Titre: Title:	Étude des performances des codes turbo
Auteur: Author:	Khaled Lajnef
Date:	2001
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Lajnef, K. (2001). Étude des performances des codes turbo [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/6953/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/6953/
Directeurs de recherche: Advisors:	
Programme: Program:	Non spécifié

UNIVERSITÉ DE MONTRÉAL

ÉTUDE DES PERFORMANCES DES CODES TURBO

KHALED LAJNEF
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES (M.Sc.A.)
(SECTION TÉLÉCOMMUNICATIONS)
JUIN 2001

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

ÉTUDE DES PERFORMANCES DES CODES TURBO

présenté par: LAJNEF Khaled

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

a été dûment accepté par le jury d'examen constitué de:

M. PIERRE Samuel, Ph.D., président

M. HACCOUN David, Ph.D., membre et directeur de recherche

M. BELZILE Jean, Ph.D., membre

À la mémoire de ma grand-mère.

Remerciements

Avant de passer en revue les différentes étapes de ce mémoire, je tiens à adresser mes vifs remerciements à tous ceux qui m'ont aidé, d'une manière ou d'une autre, à mener à bien ce travail.

En premier lieu, je tiens à exprimer ma grande reconnaissance envers mon directeur de recherche, Dr. ***David Haccoun*** pour l'aide, le soutien constant et les conseils qu'il m'a prodigués tout au long de ce travail. Je le remercie aussi pour m'avoir soutenu financièrement durant toute la période de ma recherche.

Mes remerciements vont également au Ministère de l'Education Supérieure de Tunisie pour la bourse qu'il m'a octroyée.

J'aimerais aussi remercier tous les membres du jury qui me font l'honneur de juger ce travail.

Je ne pourrais conclure sans remercier toute ma famille, mes amis et tous mes collègues de laboratoire avec qui j'ai passé des merveilleux moments: Brice, Christian, Frédéric, Guillaume, Ilham, Jérôme, Mahdi, Pierre, Pierre-Frédéric...

Résumé

La découverte des codes turbo dans la première moitié des années 90 a fortement fait évoluer le domaine du codage correcteur d'erreurs. Depuis, leurs performances ont été reconnues dans le monde académique, et ils ont d'ores et déjà été adoptés, entre autres, comme standards de transmission de INMARSAT (liaisons avec les navires), de CCSDS (liaisons en espace lointain) et pour le CDMA2000.

Le projet a pour objet principal l'étude de la performance des codes turbo en terme de probabilité d'erreur par bit paramétrée par les différentes composantes constituant ce système ainsi que la diminution du délai de processus de décodage turbo. Dans nos recherches, nous nous sommes intéressés à une nouvelle technique de perforation qui permet d'augmenter le taux de codage et d'obtenir des performances des codes turbo perforés s'approchant de la limite de Shannon.

Pour atteindre nos objectifs, l'effet du choix des codeurs élémentaires sur la performance du décodeur a été étudié. De même les divers types d'entrelacement ont été analysés et finalement le décodage itératif utilisant l'algorithme MAP a été abordé ce qui a permis de proposer une nouvelle architecture de décodage à laquelle nous nous sommes intéressés. D'autre part, une analyse particulière des codes turbo perforés a été effectuée permettant de déterminer comment doit être choisie la matrice de perforation pour certains taux de codage supérieurs à 2/3.

Le codeur/décodeur permet d'atteindre des performances proches des limites théoriques, à de faibles taux d'erreurs, pour des tailles de blocs et des rendements de codage programmables. D'ailleurs dans ce mémoire, il s'agit aussi de définir quelques principes de fonctionnement en s'appuyant sur des résultats de simulation sur ordinateur.

Abstract

The discovery of the turbo codes in the first half of the nineties have lead to remarkable progresses in the domain of error correcting codes. Since, their performances are recognized in the academic world, and they were adopted for the standards in communication systems such as INMARSAT (connections with the ships), CCSDS (Consultative Committee for Space Data Systems) and CDMA2000....

The main purpose in this project is the study of the performance of turbo codes in term of bit error rate depending on different components that constitute such coding system. In addition the reduction in the delay of turbo decoding process is also investigated. In our research, we were interested in a new puncturing technique which increases the overall coding rate and restores the performances of punctured turbo codes back to the Shannon limit.

To achieve these objectives, the effect of choosing constituent codes on the performance of turbo decoding has been investigated. In the same way, various interleaver types have been analysed and finally iterative decoding using algorithm MAP has been treated in order to propose a new decoding architecture which was studied. Furthermore, a particular analysis of the punctured turbo codes has been examined allowing us to determine how to select the punctured matrix for certain coding rates higher than 2/3.

The encoder/decoder allow to achieve performances close to the theoretical limits, at high error performances, for programmable sizes of blocks and coding rates. The new principles of operations proposed in this master's thesis are verified using computer simulations.

Table des matières

Liste des figures

Liste des sigles et des symboles

Les sigles

APP	: A Posteriori Probability
ARCH2	: Architecture 2
AWGN	: Additive White Gaussien Noise
BER	: Bit Error Rate
BCJR	: Bahl, Cocke, Jelinek and Raviv
BPSK	: Binary Phase Shift Keying
CE	: Cross Entropy
CE-P	: Cross Entropy in Parallel mode
CNS	: Convolutionnels Non Systématiques
CRS	: Convolutionnels Récursifs Systématiques
DEC1	: Décodeur 1
DEC2	: Décodeur 2
HDA	: Hard-Decision-Aided
HDA-mixed	: Hard-Decision-Aided-mixed
HDA-P	: Hard-Decision-Aided in Parallel mode
LLR	: Logarithm Likelihood Ratio
MAP	: Maximum A Posteriori
SCR	: Sign-Change-Ratio
SCR-P	: Sign-Change-Ratio in Parallel mode
SOVA	: Soft Output Viterbi Algorithm

Les symboles

K	: Longueur de contrainte d'un code convolutionnel
D	: Entier de décalage pour les entrelaceurs Convolutionnels
M	: Mémoire d'un code convolutionnel
V	: Le nombre des additionneurs modulo 2 d'un code convolutionnel
L_I	: Longueur de la réponse impulsionnelle d'un code convolutionnel
R	: Le taux de codage d'un code convolutionnel
N	: Taille d'une séquence d'information binaire
d_h	: Distance de Hamming
$d_c(n)$	: Fonction de distance des colonnes d'ordre n d'un code convolutionnel
d_{free}	: Distance libre d'un code
$T(D, B, L)$	: Fonction de transfert d'un code convolutionnel
E_b	: Énergie du signal par bit
N_0	: Densité spectrale de puissance de bruit
E_b/N_0	: Rapport signal sur bruit
$Q()$	: Fonction Q
P_B	: Probabilité d'erreur par bit
P_l	: Probabilité d'erreur pour un chemin incorrect qui diffère du chemin correct sur l symboles
$A_{w,d}$	: Fonction de distribution de poids d'un code convolutionnel
σ^2	: Variance du bruit dans le canal
$A_{w,d}^{C_p}$	: Fonction de distribution de poids d'un codeur turbo
R_{gs}	: taux de codage pour des codeurs montés en série
R_{gp}	: taux de codage pour des codeurs montés en parallèle
X_k^s	: Symbole systématique à la sortie de l'encodeur turbo
X_k^{1p}	: Premier symbole de parité à la sortie de l'encodeur turbo

Les symboles

X_k^{2p}	: Deuxième symbole de parité à la sortie de l'encodeur turbo
$\mathbf{G}_{TC}$	: Matrice génératrice de l'encodeur turbo
Λ_n	: Rapport de vraisemblance à l'instant n
$\tilde{\Lambda}_n = La_n$	: Information à priori à l'instant n
Le_n	: Information extrinsèque à l'instant n
S_n	: L'état du codeur CRS à l'instant n
$\underline{\gamma}_n^i(\mathbf{R}_n, s', s)$	: Métrique de branche à l'instant n
$\underline{\alpha}_n(s)$	: Métrique d'état en avant à l'instant n
$\underline{\beta}_n(s')$	: Métrique d'état en arrière à l'instant n
Y_k^s	: Symbole d'information systématique reçu à l'instant k
Y_k^{1p}	: Symbole de parité 1 reçu à l'instant k
Y_k^{2p}	: Symbole de parité 2 reçu à l'instant k
$T_k(i)$	: Entropie croisé du décodeur k à la $i^{\text{ème}}$ itération
$C^2(i)$	: Nombre de changement de signe entre les éléments des séquences d'information extrinsèque
$\mathbf{P}$	: Matrice de perforation

Liste des tableaux

Liste des annexes

Introduction

La théorie de l'information fondée par Claude E. Shannon en 1948, fournit un ensemble de développements théoriques qui permettent l'évaluation de la capacité des canaux de communication. Aujourd'hui, l'application de cette théorie occupe au sein des sciences et techniques de l'information une place de plus en plus importante comme le montre le nombre sans cesse croissant des systèmes de communication: téléphone sans fil, satellites de communications, etc.

Le but principal d'un système de communication numérique tel que montré à la figure 1 est de transmettre l'information avec un maximum d'efficacité et de fiabilité. Cependant, dans une telle liaison de transmission numérique, le signal reçu peut occasionnellement différer de celui transmis. Cela est dû en partie à l'existence des bruits dans les canaux de communication qui causent des erreurs de transmission. La probabilité de ces erreurs étant fonction du rapport signal à bruit. Une des

solutions pour améliorer la qualité d'un système de communication est d'augmenter la puissance d'émission. Malheureusement cette solution est souvent très coûteuse, voire même impossible à réaliser.

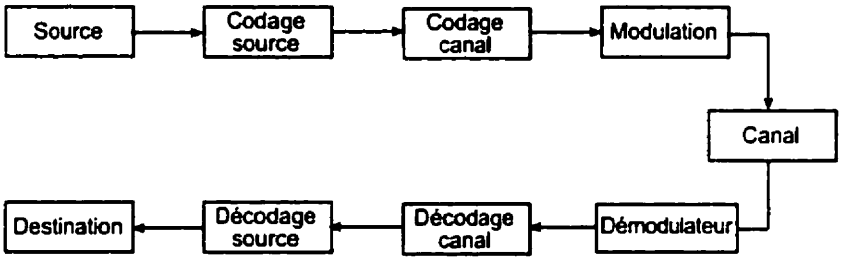


Figure 1: Diagramme d'un système de communication numérique

De nos jours, l'alternative pour augmenter la fiabilité d'une liaison de transmission, consiste à coder le message à transmettre en lui ajoutant des symboles dits de redondance selon une règle bien déterminée. Ces symboles sont nécessaires pour protéger le signal à émettre contre les erreurs de transmission. En effet, l'algorithme de décodage à la réception utilise cette redondance pour détecter, puis éventuellement, corriger les symboles erronés.

En 1948, Shannon a montré qu'il est théoriquement possible de transmettre l'information à travers un canal avec une probabilité d'erreur arbitrairement faible à condition que le débit de la transmission soit inférieur à la capacité du canal [40].

Il a démontré cette capacité pour un canal à bruit blanc gaussien additif de densité de puissance $N_0/2$ par:

$$C = W \log_2(1 + \frac{P}{N_0 W}) \text{ bits/s} \tag{1}$$

où W est la largeur de bande du canal exprimée en Hertz et P est la puissance moyenne du signal dans un intervalle de signalisation de durée T secondes.

Désignons, par R_s le débit binaire de transmission, par E_b l'énergie moyenne par bit et par $\eta = R_s/W$ le nombre moyen des bits d'information transmis durant T secondes, définie comme efficacité spectrale, d'où $\frac{P}{N_0 W} = \frac{\eta E_b}{N_0}$. A partir de (1) et sachant que $R_s < C$, la limite inférieur du rapport E_b/N_0 s'exprime par:

$$\frac{E_b}{N_0} > \frac{2^\eta - 1}{\eta} \tag{2}$$

La relation (2) s'exprime en fonction du taux de codage R [28] par:

$$\frac{E_b}{N_0} > \frac{2^{2R} - 1}{2R} \tag{3}$$

En principe, cela signifie que si on abandonne la transmission non codée au profit d'une technique avec codage, nous pourrions effectivement réaliser un canal de transmission transparent, c'est à dire un canal qui n'introduira pas de détérioration importante du signal. Malheureusement, le théorème proposé par Shannon, ne fournit aucune méthode pratique pour déterminer un tel code.

Le processus de codage dans un système de communication numérique s'effectue en deux étapes : codage de source et codage de canal (ou de transmission). Le rôle du codeur de source est d'éliminer la redondance contenue dans l'information produite par la source et de représenter celle-ci numériquement alors que le codeur de canal adapte le signal au canal et réintroduit généralement une redondance efficace sous la forme de correction d'erreurs. La fonction de correction d'erreurs (ou décodage canal) est devenue essentielle dans la conception des systèmes de télécommunication.

Dans la suite du mémoire, nous nous intéresserons essentiellement à l'aspect codage et décodage de canal du système de communication numérique. Nous ne nous intéresserons donc pas à ce qui se situe en amont du codeur de canal ou en aval du décodeur de canal.

Il existe deux grandes familles des codes correcteurs d'erreurs (codeur de canal). La première est celle des codes en blocs où une séquence de k bits d'information est codée dans un bloc de n symboles ($n > k$). La deuxième est celle des codes convolutionnels utilisés pour la transmission des symboles d'information qui arrivent en série sans structures en blocs. Dans nos travaux, la technique de codage utilisée est celle des codes convolutionnels.

Différents types d'algorithmes de décodage ont vu le jour. Nous les classerons en deux grandes catégories. Dans la première, il s'agit des algorithmes de décodage de type algébrique qui sont bien adaptés à la structure des codes en blocs. Tandis que la deuxième catégorie englobe les algorithmes qui fournissent une mesure de fiabilité sous forme probabiliste sur chaque symbole décodé. En fait, cette technique de décodage utilise explicitement les statistiques qui décrivent le comportement aléatoire du canal de communication pour tenter de déterminer quel message a été transmis. Elle est essentiellement la plus utilisée pour les codes convolutionnels.

Le but de la recherche dans le domaine de la théorie de l'information, est de trouver un système de contrôle d'erreur qui s'approchera le plus de la limite de Shannon. Pour cette raison, il faut trouver des codes ayant des bonnes propriétés de distance et déterminer leurs structures, et développer un algorithme de décodage adéquat ayant la plus faible complexité.

Plusieurs approches ont été proposées. Parmi ces solutions nous citons la notion de codage par la concaténation en série d'un code de Reed-Solomon [36] avec un code convolutionnel. Malheureusement tous les résultats des travaux qui ont été menés sur ce sujet n'ont pas réussi à s'approcher assez de la limite de Shannon.

Par ailleurs, une nouvelle famille de codes correcteurs, les codes turbo, inventée en France (ENST Bretagne, Brest) en 93, ont été capable d'offrir des gains

de codage supérieurs à 9 dB avec les taux de redondance usuels, surclassant ainsi tous les procédés de codage antérieurs. En particulier, lorsque les blocs à coder sont courts, de l'ordre de quelques centaines de bits, la solution concurrente, concaténation d'un code de Reed-Solomon et d'un code convolutionnel est fortement pénalisée. Nous pouvons même affirmer aujourd'hui que les codes turbo sont devenus importants pour la protection de données transmises par blocs courts (Internet, ATM, CDMA2000, 3GPP, UMTS entre autres). Pour d'autres applications (espace lointain, télédiffusion, ...), les codes turbo offrent également les meilleures performances. Ils ont par exemple fait l'objet d'une normalisation par le CCSDS (Consultative Committee for Space Data Systems) et les agences spatiales (NASA, ESA, ...) utiliseront les codes turbo pour toutes leurs futures missions.

De plus en plus de systèmes se définissent à l'aide de cette nouvelle famille de codes correcteurs, mais la grande souplesse dans la définition d'un code turbo, qui apparaissait initialement comme un avantage, amène aujourd'hui la multiplication des propositions techniques. Cependant, cette méthode de codage présente d'une part certaine complexité surtout au niveau de l'algorithme de décodage et d'autre part un grand délai qui dépend du nombre d'itérations du décodage itératif.

Dans ce mémoire, nous avons essentiellement présenté le principe de fonctionnement des codes turbo. Nous avons fait une analyse de performances des codes turbo selon les différentes composantes constituants ce système. Nous avons aussi examiné les délais de décodage introduits par les codes turbo. Pour améliorer ces délais, nous avons étudié l'architecture parallèle de décodage itératif turbo. De plus, nous nous sommes intéressés à une nouvelle technique de perforation qui permet d'augmenter le taux de codage et d'avoir des performances des codes turbo proches de la limite de Shannon.

0.1 Plan du mémoire

Le mémoire est organisé en cinq chapitres et une conclusion.

Le chapitre 1 introduit les propriétés des codes convolutionnels. Nous présentons les codes convolutionnels systématiques récursifs et non récursifs. Par la suite, un rappel sur le décodage de ces types des codes est proposé. Finalement, une description de la modulation BPSK et des différents types de canaux est également présentée.

Le chapitre 2 porte sur la partie codage des codes turbo. Plus particulièrement, nous présentons le principe de fonctionnement de la concaténation parallèle des codes convolutionnels séparés par des entrelaceurs. Nous abordons également les différents types des entrelaceurs utilisés en codage turbo. La borne union sur la performance d'erreur des codes turbo est aussi présentée dans ce chapitre.

Le chapitre 3 aborde le principe de décodage itératif en particulier celui des codes turbo. Après avoir présenté les différents algorithmes de décodage, une analyse de la performance des codes turbo en fonction des différentes composantes constituants ce dernier est étudiée. Dans ce chapitre nous nous intéressons aux critères d'arrêt du processus de décodage itératif des codes turbo dans le but de limiter le temps.

Au chapitre 4, Nous allons évoquer le problème de délai du décodage turbo. Nous présentons les inconvénients du mode de décodage série. Par la suite, nous proposons une solution qui consiste à utiliser l'architecture parallèle aussi bien au niveau codage que décodage. Une étude comparative de performance de cette architecture et du mode série est examinée. Finalement, les critères d'arrêt en mode parallèle sont abordés.

Le chapitre 5 est consacré à l'étude de la performance des codes turbo perforés. En effet, nous examinons l'influence d'une augmentation de taux de codage utilisant la technique de perforation sur le comportement des codes turbo. Pour certain taux de codage, nous proposons une nouvelle méthode de perforation afin d'augmenter l'efficacité de correction du code turbo.

0.2 Liste des contributions

Les contributions apportées par ce travail de recherche sont les suivantes:

- Implémentation des programmes pour différents types d'entrelaceurs: Convolutionnel, Simple décalage cyclique, Hélicoïdale, Pseudo-aléatoire pair-impair, Symétrique, Pseudo-aléatoire pair-impair symétrique, S-aléatoire.
- Création d'un nouveau type d'entrelaceur: Double décalage cyclique.
- Etude de la borne union des codes turbo et mise en place d'un nouveau algorithme pour la détermination de la distribution des poids de ces derniers.
- Etude de la performance des codes turbo suivant différents paramètres tels que le choix des codes élémentaires et le type des entrelaceurs.
- Etude comparative entre les divers critères d'arrêt qui permettent d'interrompre le processus de décodage turbo. Des modifications ont été proposées.
- Etude approfondie de l'architecture parallèle de décodage turbo avec une mise en place des différents critères d'arrêt adaptés à cet effet.
- Détermination d'une nouvelle méthode de perforation qui permet, à certains taux de codage des codes turbo, de se rapprocher encore de la limite de Shannon.

Chapitre 1

Généralités sur les codes convolutionnels

1.1 Introduction

Dans ce chapitre, nous décrivons sommairement le codage convolutionnel en nous limitant à ce qui sera utile par la suite. On fera constamment usage au cours de ce chapitre de résultats théoriques ou pratiques dont la source peut être retracée dans plusieurs ouvrages sur la théorie des communications comme [10], [34] et [47].

Nous allons examiner les critères guidant le choix d'un bon code. Par la suite, nous présenterons les codes convolutionnels récursifs systématiques (CRS) qui conduisent pour des faibles rapports signal à bruit à des probabilités d'erreur inférieures à celles des codes convolutionnels non systématiques (CNS).

Finalement, et après avoir décrit brièvement les différents types de canaux et la modulation PBSK utilisée, nous présenterons le principe de décodage des codes convolutionnels.

1.2 Les codes convolutionnels

1.2.1 Principe de base

Le codage convolutionnel est principalement introduit par Elias [18]. Son principe est illustré par le schéma de la figure 1.1.

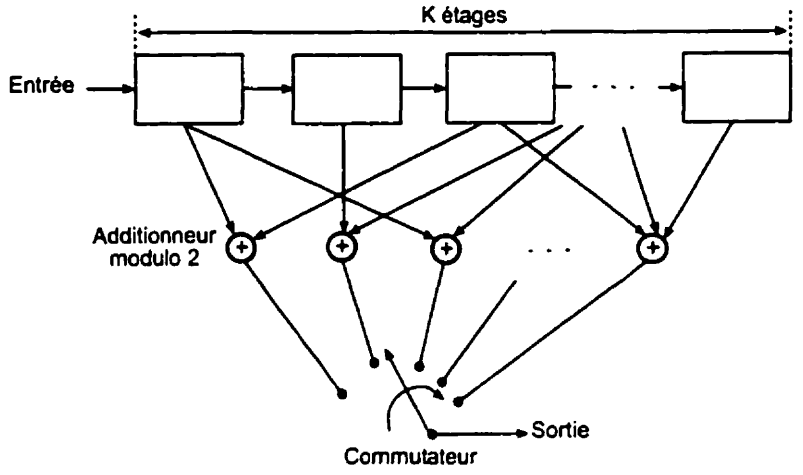


Figure 1.1: Principe d'un codeur convolutionnel

Un codeur convolutionnel de taux $1/V$ est une machine linéaire constituée d'un registre à décalage de K étages, de V additionneurs modulo 2 qui sont reliés à certaines des cellules de registre et d'un commutateur qui sélectionne séquentiellement les sorties des V additionneurs. La valeur K est appelée la longueur de contrainte du codeur.

Au début du codage, toutes les cellules de l'encodeur sont initialisées à zéro. Les bits d'information sont introduits dans le registre un par un et la sortie de l'encodeur est égale au produit de convolution entre le symbole d'information présent à l'entrée du codeur et les $(K-1)$ symboles qui précédent ce dernier. La mémoire M du codeur convolutionnel est égale à $K-1$. A la fin du codage d'une séquence de bits d'information, nous devrons réinitialiser les cellules de registre par l'insertion d'une

séquence additionnelle connue, appelée queue ou terminaison du treillis. Cette dernière est très importante pour coder correctement d'autres séquences.

Pour illustrer mieux les codes convolutionnels, nous allons présenter un exemple d'un encodeur de taux de codage $R = 1/2$ et de longueur de contrainte $K = 3$.

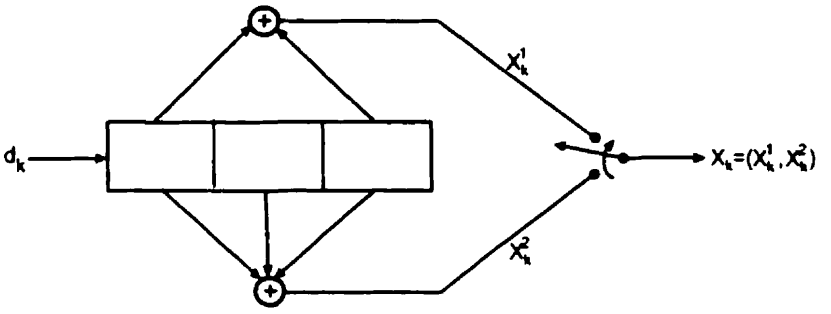


Figure 1.2: Codeur convolutionnel $R = 1/2$.

Un codeur convolutionnel est caractérisé d'une part par ses séquences génératrices (vecteurs de connexions: $\mathbf{G}_i$, $1 \leq i \leq V$) qui spécifient les connexions entre les étages de registre et les V additionneurs modulo 2 et d'autre part par sa réponse impulsionnelle **I** qui est la sortie **X** lorsque la séquence d'entrée **D** est égale à $(100000\ldots)$. Dans notre exemple de la figure 1.2, nous avons alors:

$$\begin{aligned} \mathbf{G_1} &= (g_{10}, g_{11}, g_{12}) = (101) \\ \mathbf{G_2} &= (g_{20}, g_{21}, g_{22}) = (111) \\ \mathbf{I} &= (11, 01, 11, 00, 00, \ldots) \end{aligned}$$

Et ainsi pour un bit d_k donné, nous avons l'expression des symboles de la séquence codée x_k^j présentent à la sorties de l'encodeur :

$$x_k^j = \sum_{i=0}^{K-1} \oplus\ g_{ji}\ d_{(k-i)}, \qquad j = 1, \ldots, V. \tag{1.1}$$

Avec une représentation matricielle nous avons :

$$\mathbf{X} = \mathbf{D}\ [\underline{\mathbf{G}}] \tag{1.2}$$

Où $[\underline{\mathbf{G}}]$ est la matrice Génératrice de N lignes et $(N+K-1)V$ colonnes avec N est le nombre de bit de la séquence à codée.

$$[\underline{\mathbf{G}}] = \begin{bmatrix} 11 & 01 & 11 & 00 & 00 & 00 & \dots & 00 \\ 00 & 11 & 01 & 11 & 00 & 00 & \dots & 00 \\ 00 & 00 & 11 & 01 & 11 & 00 & \dots & 00 \\ 00 & 00 & 00 & 11 & 01 & 11 & 00 & \dots \\ \vdots & \vdots & \vdots & & \ddots & \ddots & \ddots & \\ 00 & 00 & \dots & 00 & 00 & 11 & 01 & 11 \end{bmatrix} \tag{1.3}$$

En général on exprime les séquences génératrices de l'encodeur en base octale et ainsi, pour l'exemple de la figure 2.2 nous avons :

$G_1 = 5$

$G_2 = 7$

1.2.2 Représentation graphique des codes convolutionnels

Un codeur convolutionnel est une machine à états finis. Pour un taux de codage $R = 1/V$, l'état est le contenu des $K-1$ premières cellules (mémoire du codeur). Le nombre d'états possibles est égal à 2^{K-1}.

La sortie de l'encodeur dépend de son entrée et de son état présent. Après chaque introduction d'un nouveau symbole, l'état du codeur change. Le nombre de transitions entre les différents états est fini, par conséquent, il est possible de présenter graphiquement ces transitions en indiquant l'entrée et la ou les sorties du codeur.

La représentation graphique des codes convolutionnels est nécessaire pour les

algorithmes de décodage. Les plus usuelles sont la représentation sous forme d'un treillis, d'un diagramme d'état, ou encore d'un arbre. Pour mieux expliquer ces représentations considérons notre exemple de la figure 1.2.

- ***Diagramme d'état :***

  Le diagramme d'état est un graphe dont les sommets sont les états et les flèches orientées, qui relient ces derniers, représentent les différentes transitions possibles entre eux. Nous étiquetons les flèches à l'aide du doublet (symbole à l'entrée du codeur, symboles à la sortie du codeur). La figure 1.3 représente le diagramme d'état de l'exemple considéré. Le diagramme d'état

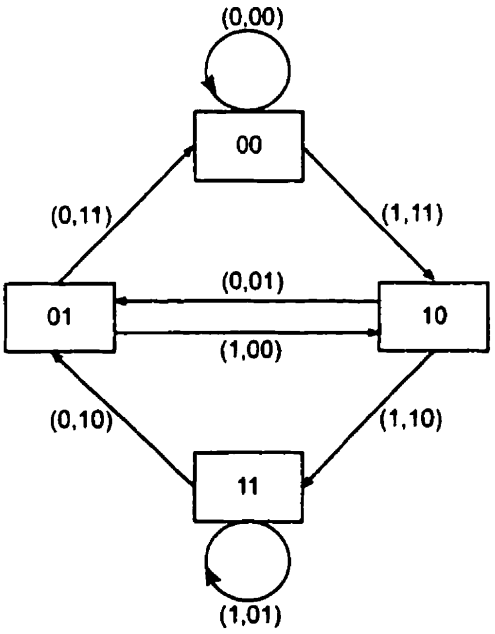


Figure 1.3: Diagramme d'état de l'encodeur de la figure 1.2

nous permet de déterminer la fonction de transfert du code. Elle est utilisée pour évaluer la distribution des poids de Hamming des différents chemins qui divergent de l'état initial (état nul) et qui reconvergent plus loin.

Comme le diagramme d'état ne nous permet pas de percevoir l'évolution dynamique du codeur dans le temps, nous utiliserons d'autres représentations telles que le graphe en arbre ou en treillis.

- ***Arbre :*** L'arbre d'encodage représente l'ensemble des séquences d'entrée-

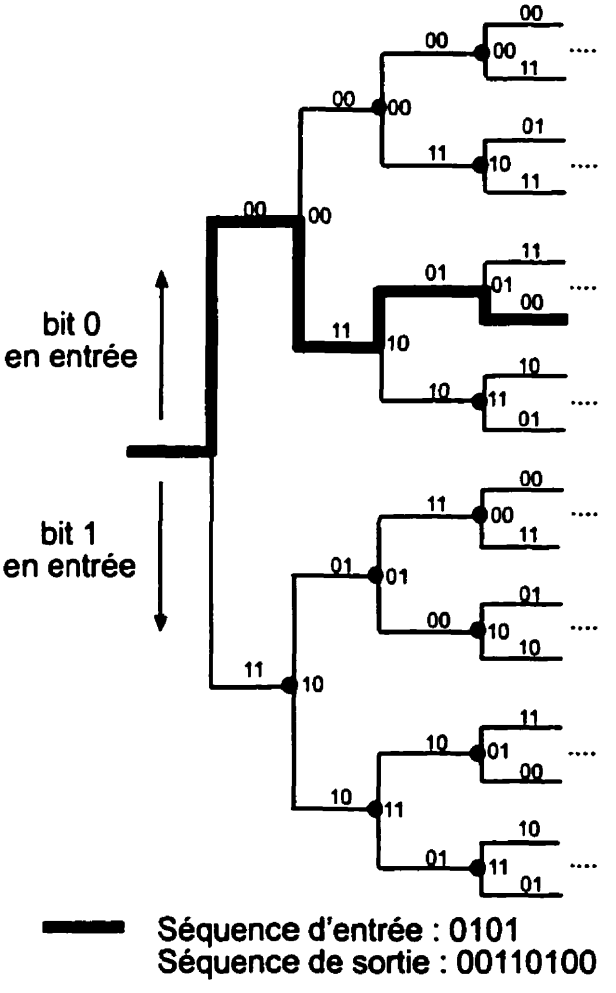


Figure 1.4: Représentation en arbre de l'encodeur de la figure 1.2

sorties possibles. Il s'agit d'un arbre binaire comme l'indique la figure 1.4. Chaque noeud de l'arbre représente un état de l'encodeur. Les noeuds sont reliés par des branches. Deux branches seulement sont issues de chaque noeud:

la branche dirigée vers la haut (respectivement vers le bas) indique que le bit d'information inséré est 0 (respectivement 1).

Les symboles codés à la sortie de l'encodeur se trouvent sur chacune des branches et à chaque nouveau bit d'information inséré, l'arbre est étendu d'un niveau vers la droite. Par conséquent, la représentation du processus d'encodage par un arbre permet de générer toutes les séquences d'information possibles à une profondeur donnée. Cependant, le nombre de noeuds dans le graphe croit exponentiellement avec le nombre de niveaux représentés (le nombre de bits insérés).

- ***Treillis :***

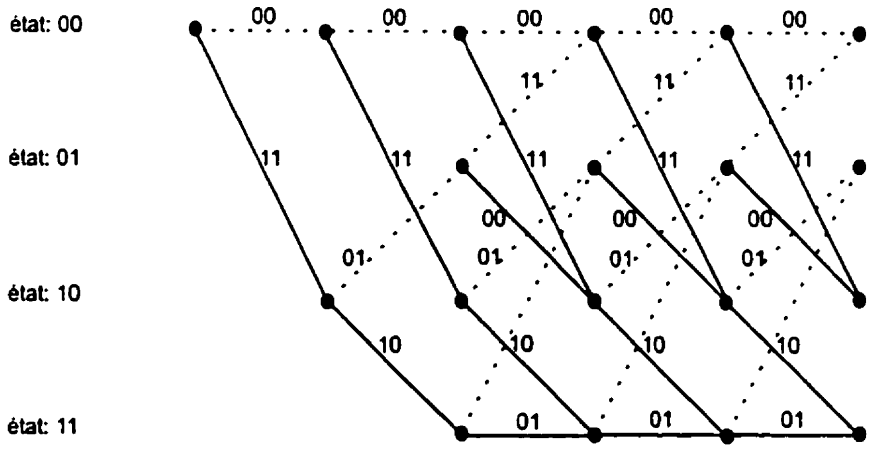


Figure 1.5: Treillis de l'encodeur de la figure 1.2

La représentation en arbre est très redondante. Le diagramme en treillis est un graphe infini dont l'ensemble des sommets (noeuds) est constitué d'une infinité de répliques des états. On n'utilise jamais le diagramme de treillis infini, mais plutôt une version tronquée, dont la dimension est de m^*N+M-1 où m : le numérateur du taux de codage et N : la longueur de la suite à coder. Les transitions entre les différents états sont représentées par des branches qui

relient les noeuds. On trouve deux branches qui sortent d'un noeud et deux autres qui y convergent.

La figure 1.5 illustre un exemple du treillis. Les branches en traits discontinus signifient que le bit inséré est 0 alors que les branches en traits continus nous informent qu'un 1 a été inséré. Les deux symboles binaires qui se trouvent près de chaque branche, représentent la séquence de sortie du codeur.

1.2.3 Choix d'un bon codeur convolutionnel

Il est très important de choisir un bon code puisque les performances du décodage en seront grandement influencées. Le problème de la recherche de bons codes convolutionnels est basé sur la recherche de la distribution des distances ou poids de Hamming des différents chemins. En fait les critères de choix les plus couramment utilisés sont: les valeurs maximales de la distance minimale et surtout la distance libre définies ci-dessous.

La distance minimale représente la distance de Hamming minimale entre les deux chemins sur la première longueur de contrainte après leur point de divergence alors que la distance libre est définie comme la plus petite distance de Hamming qui existe entre deux chemins qui divergent d'un même état initial et qui reconvergent plus loin.

Nous définissons la distance de Hamming d_h entre deux mots de code, C_1 et C_2, de longueur N, comme le nombre de symboles qui diffèrent entre eux deux. Elle est notée par :

$$d_h(C_1, C_2) = \sum_{i=1}^{N} (C_{1i} \oplus C_{2i}) \tag{1.4}$$

La fonction de distance des colonnes d'ordre n ($d_c(n)$) est la distance minimale

entre toutes les paires de mots de code de longueur n branches, issues du même point et qui divergent de ce point. Elle est donnée par :

$$d_c(n) = \min_{i \neq j}(d_h(C_i, C_j)) \tag{1.5}$$

Par conséquent la distance minimale d'un code convolutionnel est égale à:

$$d_{min} = d_c(K) \tag{1.6}$$

et la distance libre peut être exprimée par:

$$d_{free} = \lim_{n \to \infty} d_c(n) \tag{1.7}$$

La valeur de la distance libre affecte plus la capacité de correction du code. Par contre la distance minimale est plus utile pour un décodeur qui n'observe que la séquence reçue sur une longueur de contrainte comme le décodage à seuil.

Dans ce qui suit, nous nous intéresserons seulement à la distance libre puisque nous utilisons dans notre mémoire des techniques de décodage probabiliste.

La distance libre peut être obtenue à partir de la fonction de transfert du code convolutionnel qui est calculée en utilisant le diagramme d'état. Ce dernier, est redessiné en divisant l'état zéro en deux, de façon à décrire le fonctionnement exact d'un codeur convolutionnel qui diverge au début de l'état initial nul puis y reconverge. La fonction de transfert est notée comme $T(D, B, L)$ où :

D= Poids de Hamming de la branche
L= Longueur de la branche
B= Nombre de bits 1 d'information

$T(D, B, L)$ est définie par:

$$T(D, B, L) = \frac{S_f}{S_i} \tag{1.8}$$

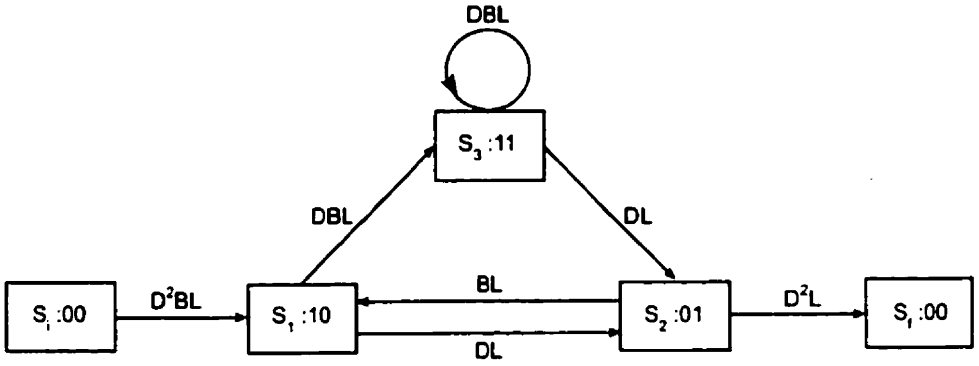


Figure 1.6: Graphe pour le calcul de la fonction de transfert de la figure 1.3

En utilisant la figure 1.6 et en se basant sur la résolution des équations suivantes:

$$\begin{aligned} S_f &= D^2 L S_2 \\ S_2 &= DLS_1 + DLS_3 \\ S_3 &= DBLS_1 + DBLS_3 \\ S_1 &= D^2 BLS_i + BLS_2 \end{aligned} \tag{1.9}$$

Nous pouvons déduire la fonction de transfert qui est :

$$\begin{aligned} T(D, B, L) &= \frac{D^5 BL^3}{(1 - DL(1 + L)B)} \\ &= D^5 BL^3 + D^6 L^4 (1 + L) B^2 + ... + D^{5+l} L^{3+l} (1 + L)^l B^{l+1} \\ &= \sum_{k=0}^{\infty} D^{5+k} L^{3+k} (1 + L)^k B^{l+k} \\ &= \sum_{k=5}^{\infty} D^k L^{k-2} (1 + L)^{k-5} B^{k-4} \end{aligned} \tag{1.10}$$

Pour $L = 1$, nous pouvons écrire :

$$T(D, B) = \sum_{k=5}^{\infty} D^k 2^{k-5} B^{k-4} \tag{1.11}$$

L'exposant de la variable D du premier terme de la série de (1.11), correspond à la distance libre du codeur qui égale dans notre exemple à 5.

L'exposant de B indique le nombre de bits 1 d'information, donc on peut estimer le nombre de bits en erreur à partir de la fonction de transfert $T(D, B)$, en transmettant une séquence de 0. Ainsi pour trouver le nombre de 1 (bits d'information), on doit dériver la fonction de transfert par rapport à B puis mettre $B = 1$, c'est-à-dire :

$$\begin{aligned} \frac{dT(D, B)}{dB}\Big|_{B=1} &= \sum_{k=0}^{\infty} D^{5+k} 2^k (k+1) \\ &= \sum_{k=d_{free}}^{\infty} D^k 2^{k-5} (k-4) \\ &= \sum_{k=d_{free}}^{\infty} c_k D^k \end{aligned} \tag{1.12}$$

avec c_k est le nombre total des bits en erreur pour tous les chemins avec le poids de Hamming égal à k.

On peut maintenant évaluer la borne supérieure de la probabilité d'erreur P_B par symbole binaire pour un canal Gaussien avec modulation BPSK par :

$$P_B \leq \frac{1}{b} \frac{dT(D, B)}{dB}\Big|_{B=1, D=P_k} \tag{1.13}$$

$$\leq \frac{1}{b} \sum_{k=d_{free}}^{\infty} c_k P_k \tag{1.14}$$

où b est le numérateur du taux de codage et où P_k la probabilité d'erreur pour un

chemin incorrect qui diffère du chemin correct sur k symboles. Elle est définie par:

$$P_k = Q(\sqrt{2kR\frac{E_b}{N_0}}), \quad k \geq d_{free} \tag{1.15}$$

En utilisant :

$$Q(\sqrt{x+y}) \leq e^{-\frac{y}{2}}\ Q(\sqrt{x}), \quad x, y \geq 0 \tag{1.16}$$

nous pouvons écrire :

$$P_k \leq e^{-(k-d_{free})R\frac{E_b}{N_0}}\ Q(\sqrt{2R\frac{E_b}{N_0}d_{free}}), \quad k \geq d_{free} \tag{1.17}$$

Maintenant la probabilité d'erreur peut être bornée par :

$$P_B \leq \frac{1}{b}\ Q(\sqrt{2R\frac{E_b}{N_0}d_{free}})\ e^{Rd_{free}\frac{E_b}{N_0}}\ \frac{dT(D,B)}{dB}\Big|_{B=1,D=Q(\sqrt{2R\frac{E_b}{N_0}k})} \tag{1.18}$$

Pour déterminer la distribution des distances des codes convolutionnels, en pratique on ne peut pas évaluer la fonction de transfert d'un tel code, plutôt on utilise des algorithmes qui explorent le diagramme en trellis ou en arbre. Cette exploration permet de déterminer le nombre des chemins qui divergent puis convergent vers l'état zéro et dont les poids de Hamming s'échelonnent de d_{free} à $d_{free} + k$. Comme sortie de cet algorithme on trouve:

$$[(d_{free}, A_{w,d_{free}}), (d_{free}+1, A_{w,d_{free}+1}), \ldots, (d_{free}+i, A_{w,d_{free}+i}), \ldots,]$$

Où $A_{w,d}$ est le nombre total des bits d'information 1 correspondant à tous les mots de code de distance de Hamming d et pour toutes les séquences binaires de poids w qui se présentent à l'entrée du codeur convolutionnel.

La borne supérieure de la probabilité d'erreur sera alors :

$$P_B \leq \sum_{d=d_{free}} B_d\ Q(\sqrt{2R\frac{E_b}{N_0}d}) \tag{1.19}$$

Où B_d est un coefficient d'erreur qui s'exprime par :

$$B_d = \sum_w \frac{1}{b} A_{w,d} \tag{1.20}$$

A partir de (1.18) ou (1.19), on peut déduire le meilleur code est celui qui à la distance libre la plus grande. Mais si on trouve plusieurs codes avec différents polynômes générateurs qui ont la même distance libre alors le meilleur est celui qui à la dérive de $T(D, B)$ ou bien la valeur B_d la plus faible.

1.3 Code convolutionnel systématique

Un codeur convolutionnel est dit systématique si les symboles codés comportent une réplique exacte des bits d'information de l'entrée. Autrement dit, un codeur systématique de taux de codage $R = 1/2$, laisse passer directement un bit d'information de l'entrée à la sortie comme symbole systématique. Les autres symboles à la sortie de l'encodeur sont appelés symboles de parités.

Les codeurs systématiques sont des codes non catastrophiques, c'est-à-dire, on ne peut trouver une boucle de poids zéro autre que la branche qui boucle sur l'état zéro dans le diagramme d'état. Cela revient du fait que la séquence d'information qui se présente à l'entrée de codeur convolutionnel systématique, se retrouve automatiquement à sa sortie.

Pour des faibles rapports signal à bruit, ces codes offrent des performances de correction d'erreurs légèrement meilleures que les codes non-systématiques. Cependant, pour des fortes valeurs du rapports signal à bruit, les codes non-systématiques sont plus performants que les codes systématiques.

1.4 Code convolutionnel récursif systématique

Une classe importante de codes convolutionnels est appelée codes convolutionnels récursifs systématiques (CRS). Un code récursif est un code convolutionnel dont un élément au moins de la matrice génératrice est une fraction rationnelle ayant pour dénominateur une fraction irréductible. Ainsi, en plus d'être systématique, les codes CRS sont caractérisés par la présence d'une boucle de retour interne qui leurs permet d'avoir la même distance libre que les codes CNS [22].

A partir d'un code convolutionnel non-systématique, on pourra construire un codeur convolutionnel récursif systématique, en conservant les mêmes séquences génératrices. Comme exemple, considérons notre codeur CNS de la figure 1.2 où le taux de codage $R = 1/2$, $K = 3$, $G_1 = 5$ et $G_2 = 7$. On peut alors trouver deux codeurs CRS de même taux $R = 1/2$, de longueur de contrainte $K = 3$ et de séquences génératrices $G_1 = 1$ et $G_2 = 7/5$ pour le code de la figure 1.7 et $G_1 = 1$ et $G_2 = 5/7$ pour le code de la figure 1.8.

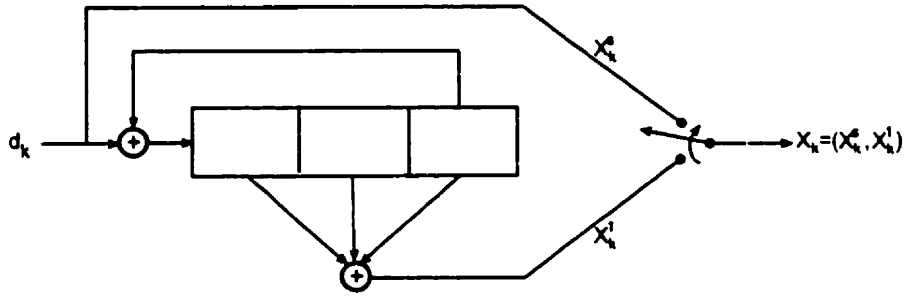


Figure 1.7: Codeur convolutionnel récursif systématique $R = 1/2, K = 3, G = [1, 7/5]$

En comparant la figure 1.9 à celle de 1.5, nous pouvons remarquer une petite différence entre les treillis de deux codeurs CRS et CNS. En fait, les deux codeurs présentent les mêmes symboles codés pour des entrées binaires différentes. Intuitivement, cela signifie d'une part que la séquence qui réinitialise le codeur CNS

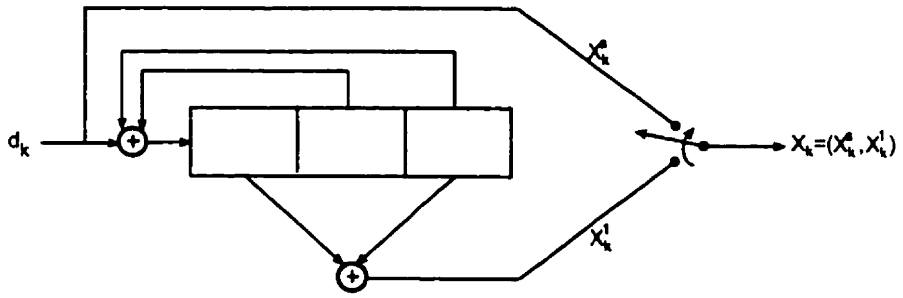


Figure 1.8: Codeur convolutionnel récursif systématique $R = 1/2, K = 3, G = [1, 5/7]$

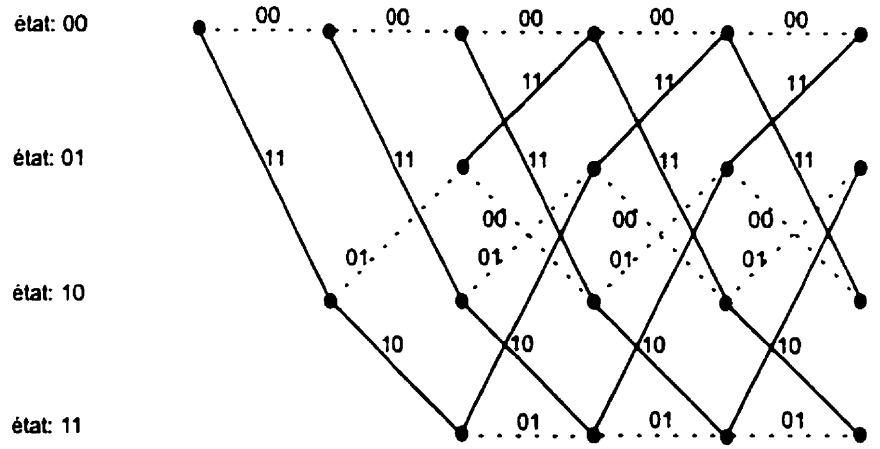


Figure 1.9: Treillis de l'encodeur de la figure 1.7

n'est pas la même pour le codeur CRS et d'autre part que le spectre de distance du premier codeur n'est pas forcément le même que celui du deuxième.

A cause de la boucle de contre réaction dans le codeur CRS, la réponse impulsionnelle d'un tel code est périodique et de période $\leq 2^{K-1} - 1$. Cela influe sur la taille de la queue qui réinitialise le codeur selon l'état final de ce dernier. Par conséquent, la queue binaire réinitialisant le codeur CRS peut être plus longue que celle avec un codeur CNS.

Pour tous les codeurs convolutionnels, la réponse impulsionnelle peut être déterminée à partir du diagramme d'état. En effet, nous avons annoncé au début de ce chapitre que la réponse impulsionnelle est la sortie du codeur pour une entrée binaire (100000...). Par conséquent et à partir du diagramme d'état, cela signifie qu'une fois le chemin diverge de l'état nul, la boucle d'entrée zéro nous donne forcément la réponse impulsionnelle. La longueur L_I de cette boucle en nombre de branches correspond à la période de la réponse impulsionnelle. Nous désignons par d_{boucle} le poids de cette boucle à entrées zéro.

Le principe est le même pour les codeurs CRS. Par exemple la figure 1.10, nous montre deux diagrammes d'état pour deux codeurs CRS de même longueur de contrainte mais avec différents vecteurs générateurs. La boucle des entrées zéro est désignée par des branches pointillées. Nous remarquons bien que la longueur L_I de cette boucle est $\leq 2^{K-1} - 1$. Pour le codeur $(1, 17/15)$, $L_I = 7$ et $d_{boucle} = 4$ et pour le codeur $(1, 15/17)$, $L_I = 4$ et $d_{boucle} = 2$.

La question qui se pose maintenant:

- Est-ce que le nombre de boucles de retour dans le codeur CRS influe sur la performance au niveau de décodage?

Nous allons répondre à cette question dans les chapitres qui viennent, en se basant seulement sur l'utilisation de ces types de codeurs convolutionnels dans le cas des codes turbo.

1.5 Canal de transmission

Le canal de transmission est divisé généralement en trois parties: modulation, canal physique de transmission et démodulation. Nous allons présenter la modulation et les modèles de canaux physiques utilisés dans nos travaux.

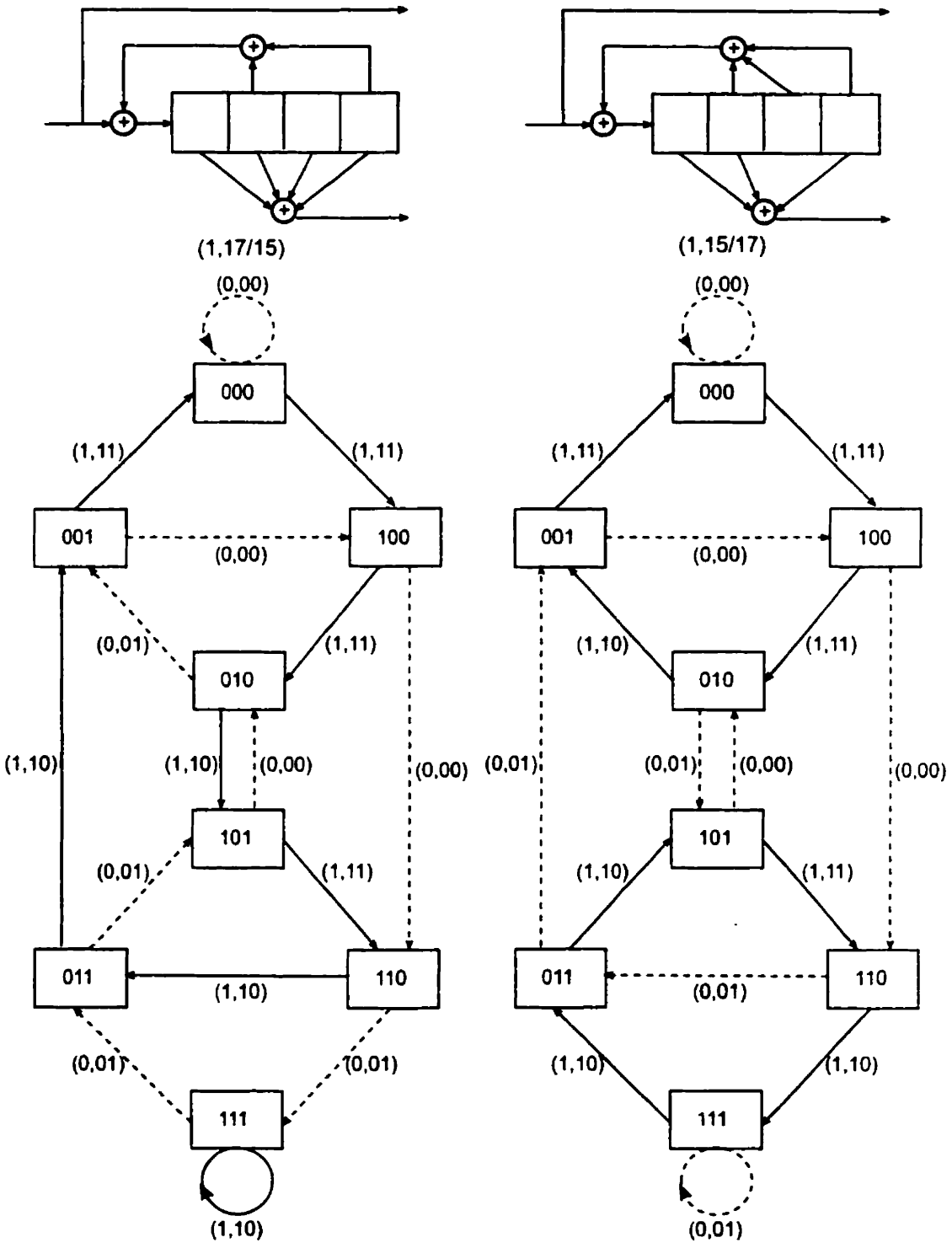


Figure 1.10: Diagrammes d'états des codeurs (1,17/15) et (1,15/17)

1.5.1 Modulation BPSK

Avant d'être transmis, la séquence multiplexée issue d'un codeur convolutionnel sera modulée en utilisant la modulation BPSK (Binary Phase Shift Keying).

Mathématiquement on peut exprimer un signal BPSK de la façon suivante:

$$\begin{aligned} s_0(t) &= -(\sqrt{2\frac{E_b}{T_b}})\cos 2\pi f_p t \qquad 0 \leq t \leq T_b \\ s_1(t) &= +(\sqrt{2\frac{E_b}{T_b}})\cos 2\pi f_p t \qquad 0 \leq t \leq T_b \end{aligned} \tag{1.21}$$

où:

$$\begin{aligned} s_0(t) &= \text{signal BPSK lorsqu'on transmet un 0} \\ s_1(t) &= \text{signal BPSK lorsqu'on transmet un 1} \\ T_b &= \text{durée de transmission d'un bit} \\ f_b &= \text{fréquence porteuse} \\ E_b &= \text{énergie par bit du signal transmis} \end{aligned}$$

1.5.2 Canal physique de transmission

Dans un système de communication, un canal physique de transmission s'occupe de l'acheminement de l'information d'un expéditeur (source) vers le destinataire (figure 1). On peut le considérer comme un milieu de propagation. Plusieurs canaux physique peuvent être trouvés : câble (torsadé, coaxial, fibre optique), guide d'onde ou l'espace entre les antennes. Ce canal n'est pas examiné du point de vue de sa réalisation matérielle, mais du point de vue de la quantité d'information qu'il peut transmettre, et du point de vue des perturbations aléatoires qu'il subit. D'ailleurs, la théorie d'information n'aurait en effet pas lieu d'être si l'on pouvait communiquer un message avec une vitesse infinie et sans bruit. Dans nos recherche nous devons utiliser des modèles idéalisés de canaux qui font abstraction de certains détails pour

faire ressortir les caractéristiques essentielles d'un tel support. Parmi les canaux les plus utilisés on peut citer: le canal à bruit blanc additif Gaussien (AWGN) et le canal de Rayleigh.

1.5.2.1 Canal AWGN :

Le bruit de canal AWGN joue un rôle absolument essentiel, en théorie comme en pratique. C'est un signal aléatoire , $n(t)$, stationnaire, Gaussien, centré, ayant une densité spectrale (symétrique) uniforme $\frac{N_0}{2}$.

L'appellation *additif* provient du fait que le signal $n(t)$ s'additionne au signal transmis. Ainsi, le signal reçu $r(t)$ s'exprime en fonction du signal transmis $x(t)$ par:

$$r(t) = x(t) + n(t) \tag{1.22}$$

Le bruit blanc gaussien n'a pas une puissance moyenne finie. Il n'a donc pas d'existence concrète. C'est une approximation du bruit thermique qui est extrêmement utilisée. La justification de cette analogie vient du fait que la densité spectrale du bruit thermique est pratiquement constante sur les supports des spectres des signaux aléatoires auxquels on a affaire. Ce type de bruit reflète bien la réalité d'un canal de transmission spatial.

1.5.2.2 Canal de Rayleigh :

Dans les canaux de transmission radiomobiles, les signaux reçus présentent des évanouissements profonds, provoqués par les phénomènes des chemins multiples et l'effet Doppler.

En fait, l'antenne à la réception reçoit non seulement le signal utile mais aussi d'autres signaux appelés chemins multiples, générés par les reflections sur différents obstacles comme les immeubles, les arbres, ... Ces signaux arrivent d'une part en retard c'est à dire avec différentes phases que le signal utile et d'autre part avec

une fréquence décalée proportionnelle à la vitesse du mobile désignée par fréquence Doppler.

Par conséquent, en théorie, le modèle du bruit AWGN ne peut pas bien représenter la variation statistique d'un tel signal. On utilise plutôt un autre modèle qui suit une loi de Rayleigh. Le signal reçu $r(t)$ s'exprime dans ce cas par:

$$r(t) = \alpha x(t) + n(t) \tag{1.23}$$

où α est l'enveloppe du signal reçu qui suit une densité de probabilité définie par:

$$p(\alpha) = \begin{cases} \frac{\alpha}{\sigma^2} e^{-\frac{\alpha^2}{2\sigma^2}} & \text{si} \quad \alpha \geq 0 \\ 0 & \text{autrement} \end{cases} \tag{1.24}$$

Cette densité est désignée par loi de Rayleigh avec une moyenne m_α et une variance notée par σ_α^2 exprimées par:

$$\begin{aligned} m_\alpha &= \sqrt{\frac{\pi}{2}}\sigma = 1.2533\sigma \\ \sigma_\alpha^2 &= (2 - \frac{\pi}{2})\sigma^2 = 0.4292\sigma^2 \end{aligned} \tag{1.25}$$

Si la fonction de densité de probabilité de (1.24) est normalisée, c'est-à-dire la puissance moyenne du signal est unitaire, alors la loi de Rayleigh deviendra:

$$p(\alpha) = \begin{cases} 2\alpha e^{-\alpha^2} & \text{si} \quad \alpha \geq 0 \\ 0 & \text{autrement} \end{cases} \tag{1.26}$$

Ainsi la moyenne m_α et la variance σ_α^2 peuvent être exprimées par:

$$\begin{aligned} m_\alpha &= 0.8862 \\ \sigma_\alpha^2 &= 0.2146 \end{aligned} \tag{1.27}$$

1.6 Décodage des codes convolutionnels

La performance d'erreur d'un système donné dépend du code et de la technique de décodage utilisée. Nous allons dans cette partie décrire le principe usuel de décodage des codes convolutionnels. En particulier nous allons présenter le décodage à maximum de vraisemblance et son efficacité pour détecter et corriger certaines erreurs introduites par le canal bruité.

Le décodage probabiliste des codes convolutionnels consiste à déterminer à partir de la séquence reçue, la séquence la plus probable. Dans ce cas le décodage utilise un critère appelé "mesure de vraisemblance" qui permet de retracer la séquence d'information. D'où le nom de décodeur à maximum de vraisemblance.

En fait, pour mieux présenter ce type de décodage, considérons une séquence d'information $\mathbf{U} = (u_1, u_2, \ldots, u_k, \ldots, u_n)$ correspondant à un message m à transmettre. Cette séquence est codée par un codeur convolutionnel de taux de codage $R = 1/v$ pour avoir en sortie:

$$\mathbf{X}^{(m)} = (\mathbf{X}_1^{(m)}, \mathbf{X}_2^{(m)}, \ldots, \mathbf{X}_n^{(m)}) = (x_{1,1}, x_{1,2}, \ldots, x_{1,v}, x_{2,1}, \ldots, x_{2,v}, \ldots, x_{n,1}, \ldots, x_{n,v})$$

Le décodeur observe la séquence bruitée:

$$\mathbf{Y} = (\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_n) = (y_{1,1}, y_{1,2}, \ldots, y_{1,v}, y_{2,1}, \ldots, y_{2,v}, \ldots, y_{n,1}, \ldots, y_{n,v})$$

à partir de laquelle on se basera pour déterminer le message $\hat{m}$ plus au moins différent de m.

Le choix de $\hat{m}$ se fait de sorte à avoir la probabilité d'erreur la plus faible en utilisant le critère de maximum de vraisemblance défini par :

$$P(\mathbf{Y}/\mathbf{X}^{(\hat{m})}) \geq P(\mathbf{Y}/\mathbf{X}^{(m')}), \qquad \forall(\hat{m} \neq m') \tag{1.28}$$

Les fonctions $P(\mathbf{Y}/\mathbf{X}^{(})$ sont appelées fonctions de vraisemblance. Alors, minimiser la probabilité d'erreur revient à maximiser ces fonctions. Dans le cas d'un canal sans mémoire la fonction de vraisemblance pour le branche i est $P(\mathbf{Y}_i/\mathbf{X}_i^{(\dot{m})})$ qui peut s'exprimer par:

$$P(\mathbf{Y}_i/\mathbf{X}_i^{(\dot{m})}) = \prod_{j=1}^{v} P(y_{i,j}/x_{i,j}^{(\dot{m})}) \tag{1.29}$$

Généralement, on utilise une métrique logarithmique pour évaluer la fonction de vraisemblance, désignée par:

$$\gamma_i = \log P(\mathbf{Y}_i/\mathbf{X}_i^{(\dot{m})}) = \sum_{j=1}^{v} \log P(y_{i,j}/x_{i,j}^{(\dot{m})}) \tag{1.30}$$

Pour la séquence d'information U, la fonction de vraisemblance s'exprime par:

$$\Gamma = \sum_{i=1}^{n} \gamma_i \tag{1.31}$$

Les valeurs $\log P(y_{i,j}/x_{i,j}^{(\dot{m})})$, γ_i et Γ sont appelées respectivement métrique de symbole, métrique de branche et métrique cumulative totale. Par la suite, le critère de maximum de vraisemblance consiste donc à choisir dans un treillis ou un arbre le chemin qui présente la métrique cumulative la plus grande.

Du fait que le nombre de chemins dans un arbre ou un treillis augmente suivant la valeur de K, la longueur de contrainte du codeur convolutionnel, la difficulté de décodage utilisant le critère de vraisemblance devient plus complexe pour des valeurs de K grandes.

Un exemple d'un algorithme de décodage qui utilise le principe de maximum de vraisemblance est l'algorithme de Viterbi [45]. Le décodage peut être optimal ou sous-optimal. A partir d'une certaine profondeur, cet algorithme est basé sur l'addition des métriques de branches à la métrique cumulative du chemin considéré, l'élimination et la comparison progressive de tous les chemins distincts aboutissant

à chaque noeud, et la sélection du chemin à métrique cumulative maximale. Ce dernier est appelé chemin survivant. Dans le cas où deux chemins ont la même métrique, le chemin survivant sera choisi au hazard. Le nombre de chemins à conserver à chaque instant est égal à 2^M. Ainsi, les besoins d'un décodeur utilisant un algorithme de Viterbi en espace mémoire sont très grands suivant la valeur de M. Toutefois, plus K augmente plus il y aura un gain de codage c'est à dire une amélioration des performances d'erreurs.

Cette technique bien qu'ayant apporté une amélioration des performances avec une grande complexité, présente la difficulté d'atteindre ou se rapprocher de la capacité de Shannon.

1.7 Conclusion

Dans ce chapitre nous avons présenté les éléments de théorie essentiels qui nous serviront comme base pour les chapitres suivants.

Dans le chapitre 2, nous présenterons les différents éléments de la partie codage du code turbo et la borne union sur la performance d'erreur de ce dernier.

Chapitre 2

Concaténation de codes

2.1 Introduction

Il est bien connu que la complexité d'un décodeur à maximum de vraisemblance croît exponentiellement avec la valeur K. Pour diminuer la complexité et augmenter le gain de codage, Forney [21] a introduit la notion de deux ou plusieurs codes concaténés en série.

La concaténation sérielle consiste à relier deux codeurs avec ou sans entrelaceur comme l'indique la figure 2.1 [44]. Les symboles à la sortie du premier codeur sont entrelacés avant de constituer les symboles à coder par le deuxième codeur. Le taux de codage global, noté R_{gs}, de la concaténation sérielle d'un codeur 1 de taux de codage R_1 et d'un codeur 2 de taux de codage R_2 est égal à :

$$R_{gs} = R_1 R_2 \tag{2.1}$$

Au niveau décodage, on trouve deux décodeurs 1 et 2 concaténés en série et séparés

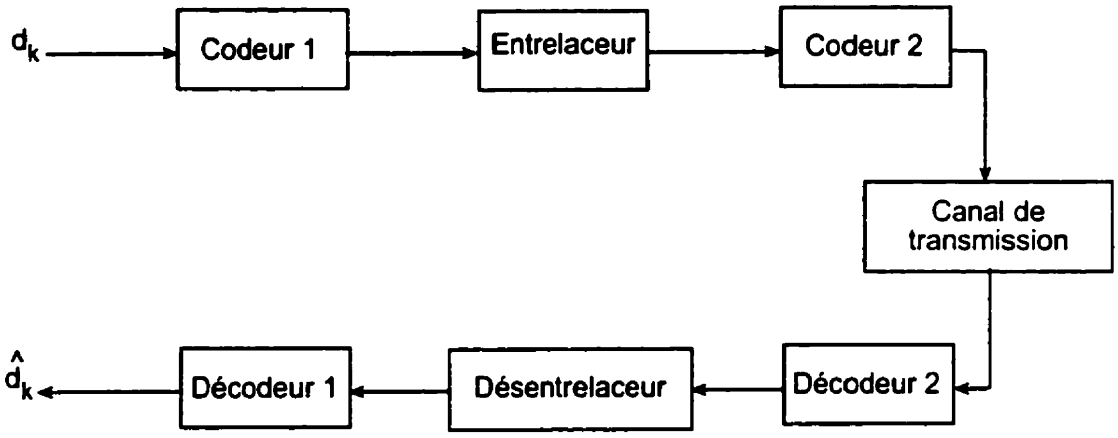


Figure 2.1: Schéma de principe d'une concaténation sérielle

par un désentrelaceur. En fait, le décodeur 2 reçoit en entrée les séquences générées par le codeur 2. Ce décodeur génère en sortie une séquence qui présente beaucoup d'erreurs groupées. Le but du désentrelaceur alors est de disperser ces erreurs pour accroître l'efficacité de correction du décodeur 1.

La concaténation sérielle des codes a été utilisée surtout dans le domaine de la communication spatiale où le premier code était un codeur convolutionnel alors que le deuxième était un codeur de type Reed-Solomon.

La partie codage du codeur turbo se base sur le même principe, c'est à dire une concaténation de deux ou plusieurs codeurs séparés par un ou plusieurs entrelaceurs. Cependant, cette concaténation se fait en parallèle et non en série. Il faudra bien remarquer que le mot turbo s'applique plutôt au niveau du décodage qu'au niveau codage. Toutefois, nous allons dans la suite désigner la partie codage du code turbo par le terme encodeur turbo.

Dans ce chapitre nous allons voir les principaux éléments qui constituent un

encodeur turbo. Plus particulièrement, nous présentons les types de codes et les différents entrelaceurs utilisés. Finalement, nous allons exposer brièvement la borne union sur la performance d'erreur du code turbo.

2.2 L'encodeur Turbo

L'encodeur turbo est constitué de deux ou plusieurs codeurs convolutionnels, récursifs, systématiques concaténés en parallèle et séparés par un ou plusieurs entrelaceurs. Cette structure a été originalement introduite par Berrou, Glavieux et Thitimajshima [8]. Le terme concaténation en parallèle provient du fait que les codeurs utilisent la même séquence d'information mais dans un ordre temporel différent alors que dans le cas de la concaténation en série, un codeur au moins doit utiliser la sortie codée de l'autre.

Nous allons nous limiter dans la suite de ce chapitre à la concaténation parallèle de seulement deux codeurs convolutionnels CRS identiques.

Le taux de codage, noté R_{gp}, de l'encodeur turbo obtenu à la suite d'une concaténation parallèle d'un premier codeur de taux de codage R_1 et d'un deuxième codeur de taux de codage R_2 vérifie la relation:

$$\frac{1}{R_{gp}} = \frac{1}{R_1} + \frac{1}{R_2} - 1 \tag{2.2}$$

soit :

$$R_{gp} = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} - 1} \tag{2.3}$$

ou encore :

$$R_{gp} = \frac{R_1 R_2}{R_1 + R_2 - R_1 R_2} \tag{2.4}$$

Pour mieux illustrer l'encodeur turbo, nous avons représenté sur la figure 2.2 un exemple d'une concaténation parallèle de deux codes élémentaires dont le taux de codage global est égal à $R_{gp} = 1/2$ après perforation. Nous trouvons dans cet exemple, en plus des composantes déjà mentionnées, comme les codeurs convolutionnels et l'entrelaceur, l'existence d'une matrice de perforation et d'une unité de multiplexage. La matrice de perforation, dont nous verrons plus clairement le rôle au chapitre 5, est très importante pour augmenter le taux de codage. Tandis que l'unité de multiplexage est utile pour multiplexer les sorties de l'encodeur turbo.

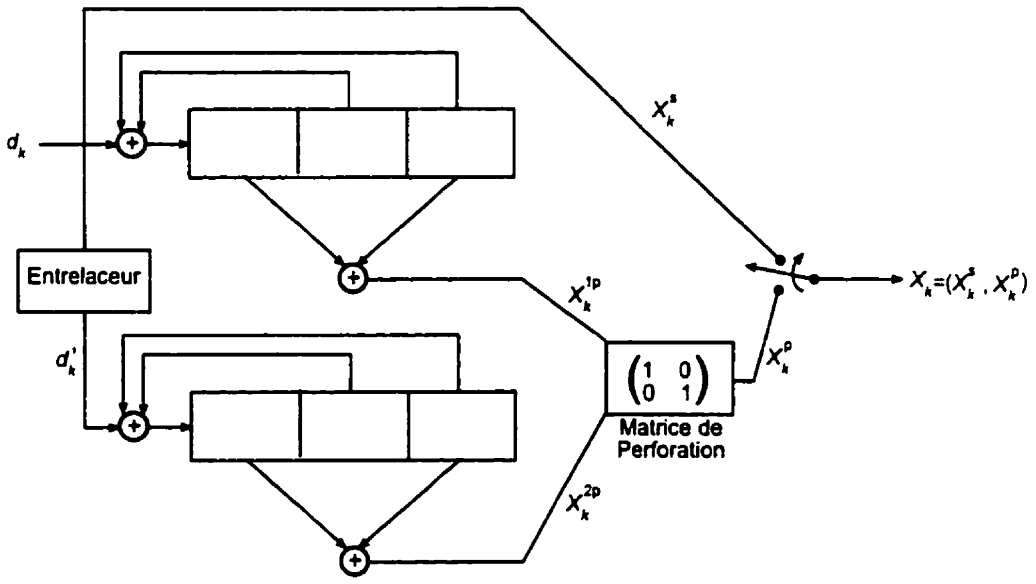


Figure 2.2: Schéma de principe d'un encodeur Turbo

En examinant la figure 2.2, un bloc de bits d'information qui se présente à l'entrée de l'encodeur turbo ($\mathbf{D} = (d_1, d_2, \ldots, d_k, \ldots, d_N)$) sera directement passé à sa sortie comme un bloc de symboles systématiques $\mathbf{X^s} = \mathbf{D} = (X_1^s, \ldots, X_k^s, \ldots, X_N^s)$. Le codeur supérieur de l'encodeur turbo reçoit les symboles d_k, dont leur ordre initial, et génère les symboles de parité X_k^{1p}, tandis que le codeur inférieur reçoit des

symboles d'_k, issues de l'entrelaceur, pour produire les symboles de parité X_k^{2p} [27]. Le taux de codage $R_{gp} = 1/2$ a été obtenu après avoir sélectionné alternativement l'un ou l'autre symbole de parité par la matrice de perforation.

2.3 L'entrelaceur

L'entrelacement est une technique très utilisée dans un grand nombre de systèmes de communications numériques. En fait, un entrelaceur prend une séquence de symboles en entrée et reproduit la même séquence à la sortie mais dans un ordre temporel complètement différent. On peut dire alors qu'un entrelaceur est un système qui permute les éléments d'une séquence, sans bien sûr, aucune répétition. On pourra se documenter au sujet des entrelaceurs en consultant l'ouvrage [12].

En premier lieu, la combinaison de deux codeurs CRS et un entrelaceur a permis de créer un code avec de meilleures propriétés de distance. D'ailleurs, il a été introduit comme un élément principal dans la concaténation des codes (en parallèle ou en série) pour pouvoir d'une part utiliser des codes avec des longueurs de contrainte K petites et d'autre part diminuer la complexité des algorithmes de décodage tout en gardant presque les mêmes performances obtenues en utilisant des codeurs classiques de grande mémoire.

En second lieu, l'entrelaceur détient un autre rôle au niveau du décodage. Il permet de décorréler les entrées des décodeurs surtout lorsque nous utilisons un algorithme de décodage itératif basé sur l'échange de l'information entre ces derniers. Autrement dit, dans le cas de décodage itératif utilisant deux décodeurs, si les séquences à l'entrée de ces derniers sont décorrélées alors il y aura une forte probabilité que les erreurs indétectables par l'un seront corrigées par l'autre. Nous détaillerons davantage le rôle de l'entrelaceur au niveau décodage itératif, dans le chapitre 3.

Il faut indiquer qu'un désentrelaceur (on l'appelle aussi délaceur) joue le même rôle qu'un entrelaceur, la différence se situe dans le fonctionnement inverse de ce dernier. Il existe différents types d'entrelaceur, mais en gros nous pouvons les subdiviser en trois familles: les entrelaceurs convolutionnels, les entrelaceurs en blocs et les entrelaceurs aléatoires.

2.3.1 Entrelaceur de type convolutionnel

2.3.1.1 Entrelaceur convolutionnel

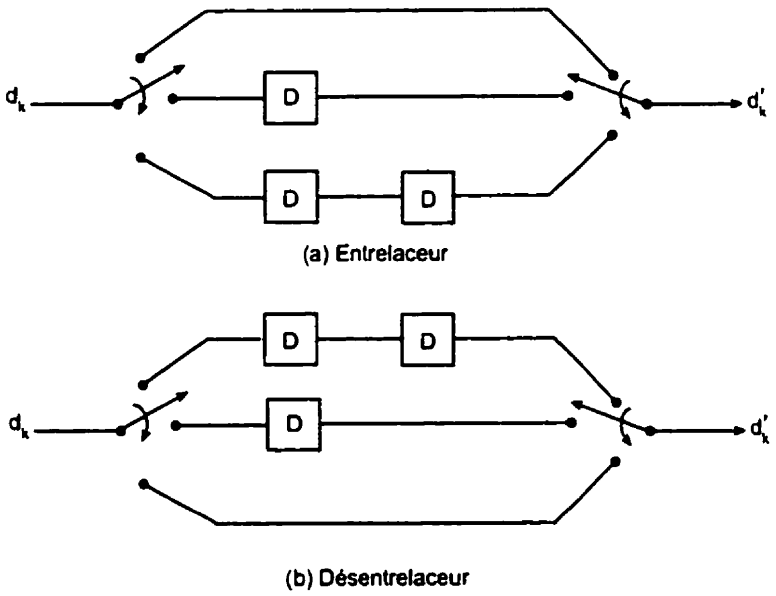


Figure 2.3: Principe d'un entrelaceur convolutionnel

L'entrelaceur convolutionnel a été introduit par Ramsey [35]. Il est constitué par des lignes de retard (ou des registres à décalages) et des commutateurs comme l'indique la figure 2.3. Les commutateurs sont placés en amont et en aval des lignes de retard afin de réordonner les positions des symboles de la séquence qui se présente à l'entrée de l'entrelaceur.

La différence de délai entre deux lignes successives est égale à D. Ainsi, le symbole qui se présente à la $i^{ème}$ ligne sera retardé d'un délai égal à iD.

2.3.1.2 Entrelaceur à simple décalage cyclique

La méthode alternative pour générer un entrelaceur convolutionnel est basée sur le décalage cyclique. Le nom initial de ce type d'entrelaceur est entrelaceur à décalage cyclique. Nous avons introduit le mot simple pour pouvoir le différencier de celui que nous allons présenter dans le paragraphe suivant. La séquence d'information est écrite dans une matrice colonne par colonne. Cette dernière doit avoir m lignes et n colonnes telles que $m \leq n$. Chaque ligne i est décalée cycliquement vers la gauche de $(i-1)D$, où $D \leq \frac{n}{m}$ est la partie entière de $\frac{n}{m}$. La lecture de la nouvelle matrice obtenue se fait colonne par colonne.

Le principe du désentrelaceur est analogue. Il faut simplement faire le décalage vers la droite au lieu de le faire vers la gauche.

Un exemple d'un entrelaceur à simple décalage cyclique est présenté à la figure 2.4 où les paramètres m, n et D sont respectivement égaux à 3, 7, et 2.

$$\begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 \end{bmatrix} \xrightarrow{\text{décalage cyclique}} \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 \\ 8 & 11 & 14 & 17 & 20 & 2 & 5 \\ 15 & 18 & 21 & 3 & 6 & 9 & 12 \end{bmatrix}$$

Figure 2.4: Entrelaceur à simple décalage cyclique

2.3.1.3 Entrelaceur à double décalage cyclique

Le principe de l'entrelaceur à double décalage cyclique est presque le même que celui à simple décalage. L'idée est de décaler non seulement les lignes mais encore les colonnes pour avoir plus de séparation entre les éléments successifs. Nous avons

trouvé plusieurs façons pour décaler les colonnes. La meilleure est celle qui utilise la parité du numéro des colonnes. En effet, chaque colonne i est décalée de $i \times D$ $mod(m)$ où D est un entier $\leq \frac{n}{m}$. Cependant, si i est pair alors le décalage se fait vers le bas plutôt que vers le haut.

Pour mieux expliquer l'entrelaceur à double décalage cyclique, nous allons considérer l'exemple du paragraphe précédent où $m = 3$, $n = 7$ et $D = 2$. La figure 2.5 illustre le résultat obtenu en utilisant l'entrelaceur à double décalage cyclique.

$$\begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 \end{bmatrix} \begin{array}{c} \text{double décalage} \\ \longrightarrow \\ \text{cyclique} \end{array} \begin{bmatrix} 15 & 18 & 7 & 17 & 20 & 16 & 12 \\ 1 & 4 & 14 & 3 & 6 & 2 & 19 \\ 8 & 11 & 21 & 10 & 13 & 9 & 5 \end{bmatrix}$$

Figure 2.5: Entrelaceur à double décalage cyclique

2.3.2 Entrelaceur de type bloc

Ce type d'entrelaceur est caractérisé par le traitement en bloc plutôt qu'en série (comme dans le cas des entrelaceurs convolutionnels) des symboles d'une séquence à réordonner. Il existe différentes catégories des entrelaceurs en bloc:

2.3.2.1 Entrelaceur bloc classique

Dans ce type d'entrelaceur, il s'agit d'écrire les symboles d'un bloc à entrelacer ligne par ligne dans une matrice, et de les lire en sortie colonne par colonne comme l'indique la figure 2.6. Il y a différentes façons pour lire des symboles dans une matrice, de gauche vers la droite (GD) ou de droite vers la gauche (DG) pour les colonnes et de haut vers le bas (HB) ou de bas vers le haut (BH) pour les lignes. Ainsi, il y a quatre façons qui sont illustrées à la figure 2.7.

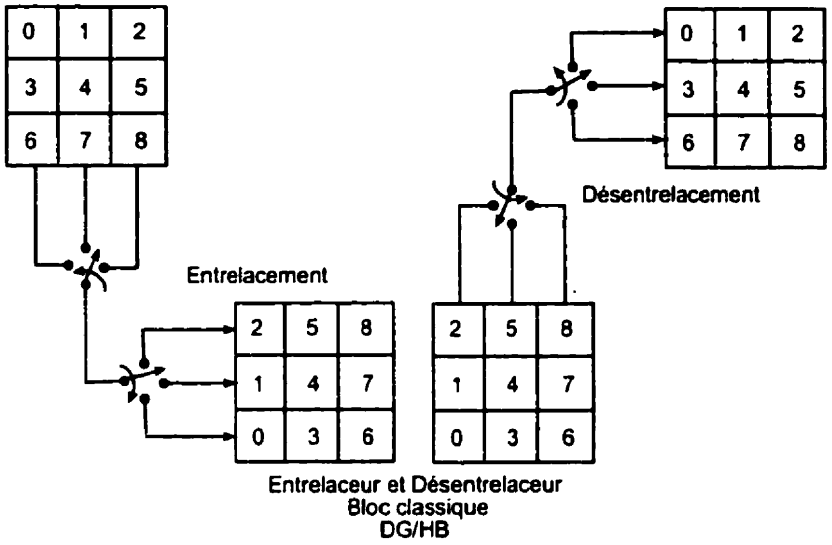


Figure 2.6: Principe d'un entrelaceur bloc classique

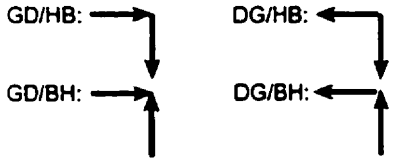


Figure 2.7: Mode d'opération pour réordonner des symboles

2.3.2.2 Entrelaceur hélicoïdal

Comme son nom l'indique, il s'agit de lire les symboles d'une matrice de façon hélicoïdale. Autrement dit, la lecture doit même se faire en suivant les diagonales de la matrice. Dans ce type d'entrelaceur, nous disposons de plusieurs façons pour changer le séquencement des symboles d'une matrice. Tout dépend par où nous commençons la lecture de la matrice. Pour mieux expliquer, nous allons nous baser sur des figures et par la suite nous présenterons nos commentaires.

Nous remarquons à partir des figures 2.8 et 2.9 que les symboles en sortie du premier type d'entrelaceur sont moins espacés que ceux du deuxième. Intuitive-

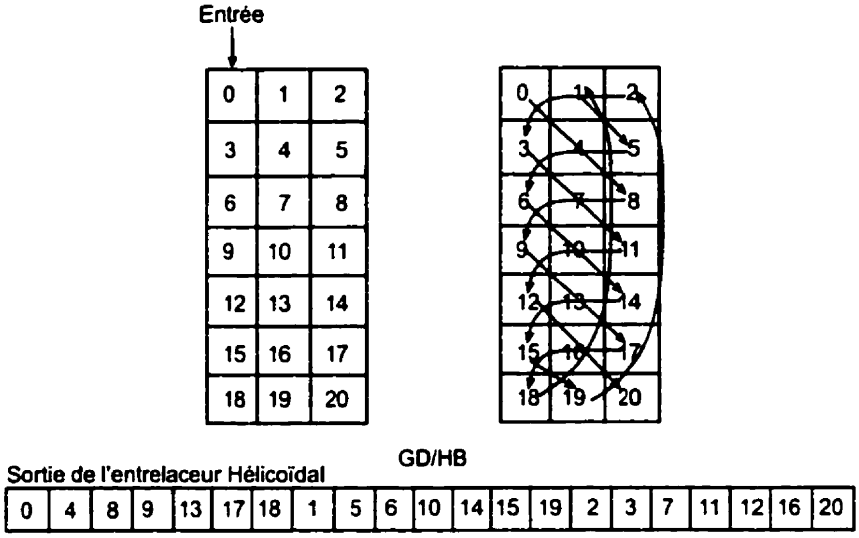


Figure 2.8: Principe d'un entrelaceur hélicoïdal GD/HB

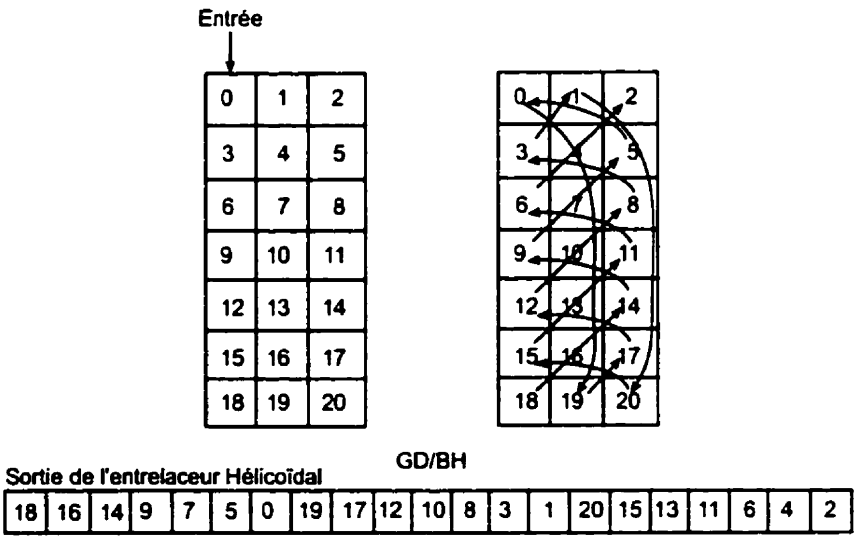


Figure 2.9: Entrelaceur hélicoïdal GD/BH

ment, nous pouvons dire que l'utilisation de deuxième type d'entrelaceur dans le code turbo devrait fournir de meilleures performances.

2.3.3 Entrelaceur de type aléatoire

2.3.3.1 Entrelaceur pseudo-aléatoire

Le concept fondamental d'un entrelaceur pseudo-aléatoire est simple mais sa réalisation en pratique est plus complexe que celle des entrelaceurs bloc classique et hélicoïdal. Une fois que les symboles d'un bloc sont introduits dans la matrice d'un entrelaceur pseudo-aléatoire, les symboles en sortie sont choisis d'une façon quasi-aléatoire de telle manière qu'il ne faut pas répéter le même symbole déjà sélectionné.

Comme la sélection est aléatoire, il sera impossible de connaître les positions des symboles à la sortie de la matrice d'entrelacement. Par conséquent, il serait utile de garder une table de correspondance entre les anciennes et les nouvelles positions des symboles entrelacés afin de pouvoir délacer ces derniers [39].

2.3.3.2 Entrelaceur pseudo-aléatoire pair-impair

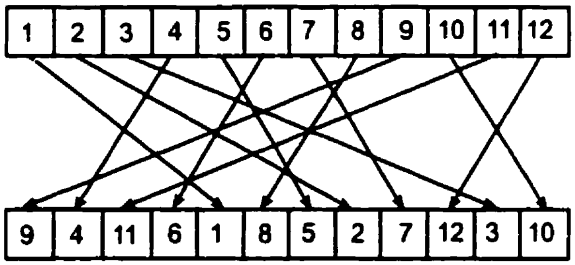


Figure 2.10: Entrelaceur pseudo-aléatoire pair-impair

Les entrelaceurs pseudo-aléatoires pair-impair sont des entrelaceurs pseudo-aléatoires avec une simple condition : les symboles avec leurs positions initiales

paires (respectivement impaires) seront entrelacés aléatoirement en sortie dans des positions paires (respectivement impaires). La figure 2.10 montre un exemple d'entrelaceur pseudo-aléatoire pair-impair.

2.3.3.3 Entrelaceur symétrique

Dans la plupart des cas, l'entrelaceur et le désentrelaceur sont implémentés séparément puisqu'ils sont différents. Par conséquent, le besoin en espace mémoire risque d'être énorme surtout pour des entrelaceurs de grandes tailles. Ce problème peut être résolu par l'utilisation d'un entrelaceur symétrique [39].

En effet, l'entrelaceur symétrique échange les positions des symboles d'une séquence deux à deux sans répétition. Autrement dit, pour deux symboles a et b de positions initiales respectives I et J, si a a la position J après entrelacement alors nécessairement b devra occuper celle de I. Un exemple illustrant le principe de l'entrelaceur symétrique est présenté à la figure 2.11.

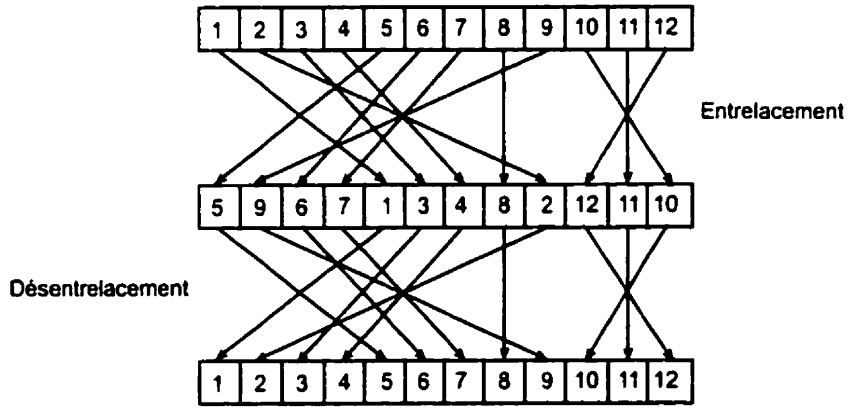


Figure 2.11: Entrelaceur et désentrelaceur symétrique

2.3.3.4 Entrelaceur pseudo-aléatoire pair-impair symétrique

Une idée intéressante serait d'ajouter une contrainte supplémentaire à l'entrelaceur pseudo-aléatoire pair-impair. Elle consiste à introduire le critère de symétrie.

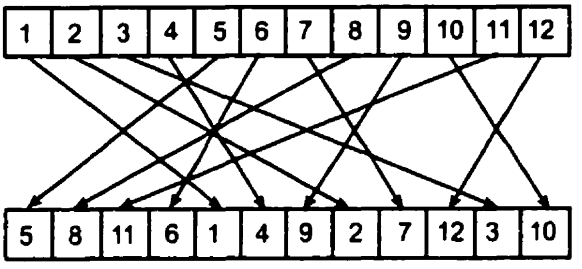


Figure 2.12: Entrelaceur pseudo-aléatoire pair-impair symétrique

Autrement dit, l'entrelaceur pseudo-aléatoire pair-impair symétrique permet d'échanger les positions, paires ou bien impaires, de deux symboles entre eux. La figure 2.12 montre un exemple d'entrelaceur pseudo-aléatoire pair-impair symétrique.

2.3.3.5 Entrelaceur S-aléatoire

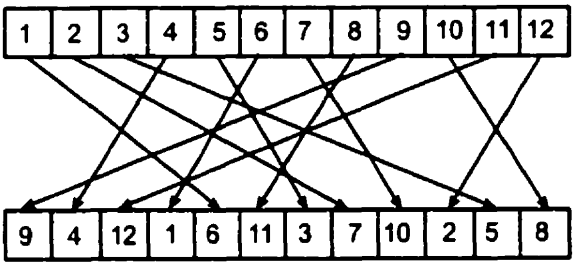


Figure 2.13: Entrelaceur S-aléatoire $S = 2$

Les entrelaceurs S-aléatoires ont été proposés pour la première fois par Divsalar et Pollar [14]. Ils permettent de changer aléatoirement le séquencement d'un bloc de telle manière que leurs symboles restent espacés d'au moins une distance S. Autrement dit, chaque position d'un symbole sélectionné sera comparé aux S positions déjà sélectionnées précédemment. Si cette position est distante d'au moins $\pm S$ par rapport aux S positions précédentes alors le symbole est retenu, sinon il est rejeté. Cela se fait jusqu'à la sélection de tous les symboles. Généralement, $S < \sqrt{\frac{N}{2}}$, où N est la longueur du bloc à entrelacer. Ainsi, pour $S = 1$ on retrouve

l'entrelaceur pseudo-aléatoire classique.

L'exemple de la figure 2.13 représente l'entrelaceur S-aléatoire pour un bloc de longueur 12, c'est-à-dire $S = 2$.

2.4 Représentation matricielle de l'encodeur Turbo

Avant d'entamer l'analyse de la performance d'erreur des codes turbo par la borne union, il est bon de considérer un encodeur turbo sous forme de code en bloc. Pour ce faire, il faudra trouver la matrice génératrice de deux codeurs convolutionnels montés en parallèle et séparés par un entrelaceur.

Désignons par **I**, **P** et **E** respectivement la matrice identité correspondant aux symboles systématiques, la matrice génératrice du codeur convolutionnel constituant l'encodeur turbo et la matrice de l'entrelaceur. La taille de ces trois matrices est $N \times N$ où N est la longueur du bloc à coder.

En fait, la matrice **E** est une matrice de permutation des symboles à coder par le deuxième codeur convolutionnel. Donc, sur chaque ligne et sur chaque colonne de cette matrice, nous devons trouver un seul 1 et tous les autres éléments seront à 0. En d'autres termes, le 1 dans la ligne i et la colonne j de la matrice E nous indique que le $i^{\text{ème}}$ symbole de la séquence à entrelacer va se trouver dans la $j^{\text{ème}}$ position de la séquence de sortie. Pour avoir plus de renseignements sur la représentation matricielle des entrelaceurs, le lecteur est invité à consulter l'annexe I.

Finalement, la matrice génératrice $\mathbf{G}_{TC}$ de l'encodeur turbo peut s'exprimer par:

$$\mathbf{G}_{TC} = [\mathbf{I}|\mathbf{P}|\mathbf{EP}] \tag{2.5}$$

où

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{E} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (2.6)$$

2.5 Borne d'union sur la probabilité d'erreur du code turbo

Comme nous l'avons vu au chapitre 1 la borne d'union sur la probabilité d'erreur pour un décodeur basé sur l'algorithme à maximum de vraisemblance dépend de la distribution des poids du codeur en question (nombre de chemins qui divergent puis reconvergent). De plus, nous avons vu au paragraphe 2.4 que le code turbo peut être représenté sous forme d'un code bloc. Par conséquent, on peut déterminer la borne d'union du codeur turbo de la même façon qu'au chapitre 1. Il nous reste à déterminer seulement la distribution des poids du code turbo.

Dans le cas du code turbo, le poids de Hamming d'un mot de code dépend des poids de la séquence des symboles systématiques, de la première séquence de parité et de la deuxième séquence de parité [17].

Soient w, z_1 et z_2 les poids respectifs de la séquence des symboles systématiques, de la première séquence de parité et de la deuxième séquence de parité. Alors, la

distance de Hamming d'un tel mot de code dans le cas du code turbo est égale à:

$$d = w + z_1 + z_2 \tag{2.7}$$

En fait, le poids de la deuxième séquence de parité est un peu difficile à déterminer. Non seulement, il dépend de w mais aussi du type et de la taille de l'entrelaceur. Par la suite, nous allons essayer d'exposer les solutions et les algorithmes permettant de déterminer la distribution des poids de code turbo.

2.5.1 Algorithme de Benedetto et al

La solution proposée par Benedetto et Montorsi [6] et adoptée par [50], consiste à utiliser un entrelaceur uniforme. En fait, en utilisant un entrelaceur de taille N, il y aura $N!$ permutations possibles avec une probabilité de $1/N!$ chacune pour réordonner une séquence de longueur N. Par contre pour une séquence de même taille mais de poids de Hamming w, il existe $\binom{N}{w}$ possibilités équiprobables chacune de probabilité:

$$\frac{1}{\binom{N}{w}} = \frac{w!(N-w)!}{N!} \tag{2.8}$$

L'utilisation d'un entrelaceur uniforme permet d'assurer une indépendance entre la distribution des poids du premier codeur et celle du deuxième. Par conséquent, on peut évaluer facilement la fonction de distribution des poids du code turbo par:

$$A_{w,d}^{C_p} = \frac{A_{w,d_1}^{C_1} \cdot A_{w,d_2}^{C_2}}{\binom{N}{w}} \tag{2.9}$$

où $A_{w,d_1}^{C_1}$ et $A_{w,d_2}^{C_2}$ sont respectivement les fonctions qui déterminent la distribution des poids pour toutes les séquences binaires de poids w du premier et du deuxième codeur de telle sorte que $d = d_1 + d_2$.

Normalement à partir d'une telle méthode nous ne pouvons pas déterminer la distance libre du code turbo. Dans [6] et [16], on trouve une approximation de la distance libre appelée distance libre effective définie par :

$$d_{free,effec} = 2 + 2z_{min} \tag{2.10}$$

où z_{min} est la distance de Hamming minimale d'un mot de code pour une séquence à coder de poids $w = 2$.

2.5.2 Algorithme utilisé pour la détermination du spectre du poids des codes turbo

L'algorithme de Benedetto détermine d'une façon globale et approximative le spectre du poids des codes turbo. Nous proposons un autre algorithme qui détermine précisément ce spectre pour un entrelaceur spécifique et qui reflète bien le mode de fonctionnement de la concaténation parallèle.

Le concept de la nouvelle solution est extrêmement simple. Il s'agit d'un algorithme qui génère toutes les séquences avec un poids w de telle sorte que le premier symbole est toujours à 1 pour assurer la divergence des chemins. Par la suite, chaque séquence sera codée par le premier et le deuxième codeur CRS. A la sortie de l'encodeur turbo, notre algorithme réalise une énumération exhaustive, il calcule en plus de la distance de la séquence systématique, les distances des séquences codées ou de parité qui convergent vers l'état zéro pour les deux CRS. Nous remarquons alors que notre algorithme tient compte du type d'entrelaceur.

La distance libre du code turbo depend d'une part de la taille et du type d'entrelaceur et d'autre part de la mémoire et la matrice génératrice du codeur CRS en question. Nous avons remarqué que la valeur de d_{free} n'est pas toujours déterminée à partir d'une séquence de poids $w = 2$. Elle peut être même évaluée pour $w = 3, 4, 5$.

Notre algorithme explore le diagramme d'état afin de déterminer le nombre total de chemins qui divergent puis convergent. Il peut déterminer le spectre en distance

du code turbo ($A_{w,d}^{C_p}$) pour toutes les séquences de poids w variant de 2 à 20. Nous avons suggéré que le nombre maximal 20 est suffisant puisque nous utilisons des séquences d'informations de longueur 20 bits. Il faut signaler aussi que le spectre est évalué seulement pour les distances entre d_{free} et $d_{free} + 10$.

L'effort de calcul et le temps d'exécution de notre algorithme est énorme surtout pour déterminer le spectre d'une séquence d'information de taille $n > 100$. Cela est dû à la génération de toutes les séquences de taille n et de poids w variant entre 2 et n. Par conséquent, la méthode de Benedetto *et al* est moins complexe et meilleure pour des séquences de taille supérieure à 100.

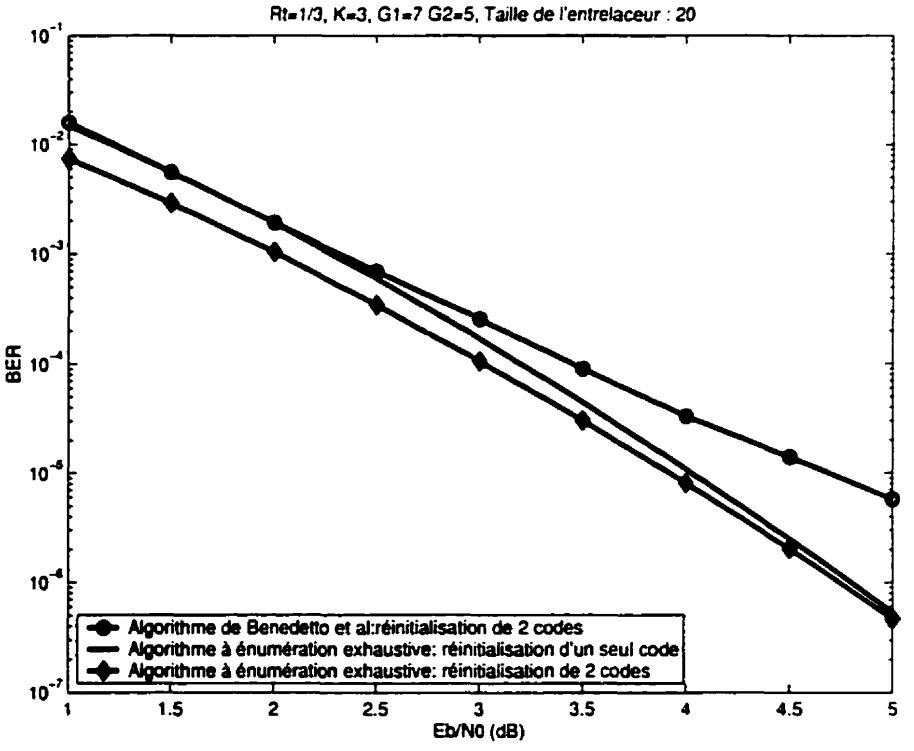


Figure 2.14: Borne union du code turbo avec l'algorithme de Benedetto et notre algorithme à énumération exhaustive pour un entrelaceur de taille 20

L'algorithme proposé calcul seulement les chemins qui divergent puis reconver-

gent une seule fois. Cela veut dire qu'il ne compte pas les chemins qui divergent.

En utilisant (1.19), la borne d'union du code turbo peut être déterminée en prenant B_d égale à:

$$B_d = \sum_w \frac{w}{N} A_w^{C_p} \tag{2.11}$$

par

$$P_B \leq \sum_{d=d_{free}} B_d \, Q(\sqrt{2R\frac{E_b}{N_0}d}) \tag{2.12}$$

2.6 Conclusion

Ce chapitre a présenté en premier lieu les constituants et l'architecture de l'encodeur turbo. Par la suite, nous avons décrit l'entrelaceur qui est un des mots-clé du code turbo. Différents types d'entrelaceur ont été présentés. Enfin, la borne union du code turbo a été évoquée afin d'avoir une idée d'une part sur la distribution de poids d'un tel code et d'autre part sur sa performance de correction.

Chapitre 3

Décodage Turbo

3.1 Introduction

Un des concepts fondamentaux du codage turbo, que nous venons de voir dans le chapitre 2, est la concaténation en parallèle de deux codeurs systématiques, récursifs (CRS) séparés par un entrelaceur.

Nous allons dans ce chapitre, nous intéresser à un autre concept qui est le décodage itératif. Après avoir exposé les principes et avant d'aborder le critère d'arrêt des itérations, nous présenterons l'algorithme de décodage le plus utilisé dans les décodeurs turbo.

Nous limitons notre étude, dans ce chapitre, au décodage turbo correspondant à la concaténation parallèle de deux codeurs.

3.2 Principe de décodage itératif

La figure 3.1 montre un diagramme simplifié d'un décodeur itératif qui est constitué de deux décodeurs (correspondant chacun à son propre codeur), d'un entrelaceur et d'un désentrelaceur.

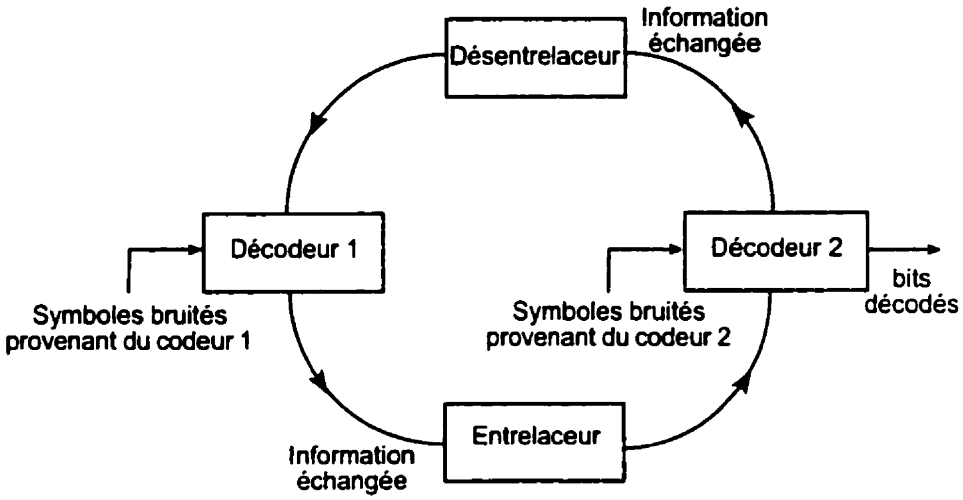


Figure 3.1: Schéma de principe du décodage itératif

Le principe de décodage itératif est très simple. Il s'agit de répéter plusieurs fois l'échange d'information pondérée sur les bits d'information entre les deux décodeurs jusqu'à l'obtention des performances de décodage désirées. En fait, le décodage itératif se fait en plusieurs itérations. Chaque itération de décodage implique deux décodeurs.

Pour une première itération et après avoir décodé les symboles associés au premier codeur, le décodeur 1 fait passer un bloc d'information en quantification douce vers le décodeur 2. Ce dernier dispose de cette information ainsi que des symboles bruités issus du deuxième codeur pour pouvoir corriger certaines erreurs qui n'ont pas été détectées par le décodeur 1. A la deuxième itération, si le décodeur 1 peut avoir accès à plus d'informations alors sûrement il fournira des performances

meilleures que celles obtenues durant la première étape. Autrement dit, le décodeur 1 reçoit en plus des mêmes symboles bruités issus du premier codeur, un bloc d'information en quantification douce provenant du décodeur 2.

Ainsi, d'une itération à une autre, le processus d'échange d'information se répète jusqu'à une saturation où les deux décodeurs ne peuvent améliorer la performance d'erreur.

La séquence d'information supplémentaire échangée qui se présente à l'entrée de l'un des deux décodeurs s'appelle séquence d'information à priori. Elle doit être indépendante de toutes les entrées de l'autre décodeur conditionnellement sur les bits d'information. En plus, elle doit contenir des symboles erronés qui sont réordonnés et non regroupés ensemble afin d'augmenter l'efficacité de décodage. D'où l'intérêt de la présence de l'entrelaceur ou de celle du désentrelaceur entre les deux décodeurs.

Généralement, les deux décodeurs utilisent un algorithme de décodage à décision pondérée, c'est-à-dire minimisant la probabilité d'erreur par symbole. Un tel décodeur est appelé décodeur APP. Il fournit une probabilité à posteriori (APP: A Posteriori Probability) sur chaque symbole d'information émis conditionnellement à la séquence reçue (les observables).

Pour illustrer les décodeurs APP, nous considérons l'exemple d'un codeur CRS de taux de codage $R = 1/2$. A la réception, nous avons deux séquences de N symboles reçues : $\mathbf{X}_1^N = (x_1, x_2, \ldots, x_n, \ldots, x_N)$ et $\mathbf{Y}_1^N = (y_1, y_2, \ldots, y_n, \ldots, y_N)$ associées respectivement à la séquence systématique et à celle de parité. Ainsi, les entrées du décodeur APP sont $\mathbf{X}_1^N$, $\mathbf{Y}_1^N$ et $\tilde{\mathbf{\Lambda}}_1^N$, la séquence de l'information à priori. Ces séquences seront désignées par:

$$\mathbf{R}_1^N = (\mathbf{X}_1^N, \mathbf{Y}_1^N, \tilde{\mathbf{\Lambda}}_1^N) \tag{3.1}$$

où $\tilde{\mathbf{\Lambda}}_1^N = (\tilde{\Lambda}_1, \tilde{\Lambda}_2, \ldots, \tilde{\Lambda}_n, \ldots, \tilde{\Lambda}_N)$ tel que chaque élément est représenté par:

$$\tilde{\Lambda}_n = \ln \frac{Pr(u_n = 1)}{Pr(u_n = 0)} \tag{3.2}$$

où u_n est le symbole d'information à l'instant n.

A la sortie du décodeur APP, la mesure de fiabilité sur chaque symbole u_n, appelée aussi rapport de vraisemblance, est notée par:

$$\Lambda_n = \ln \frac{Pr(u_n = 1|\mathbf{R}_1^N)}{Pr(u_n = 0|\mathbf{R}_1^N)} \qquad n = 1, 2, \ldots, n, \ldots, N. \tag{3.3}$$

La règle de décision qui détermine le symbole le plus vraisemblable est la suivante:

$$\begin{aligned} \Lambda_n &\geq 0 : \text{ le bit décodé est } \hat{u}_n = 1 \\ \Lambda_n &< 0 : \text{ le bit décodé est } \hat{u}_n = 0 \end{aligned} \tag{3.4}$$

Revenons maintenant au décodage itératif. Nous avons vu précédemment que $\tilde{\mathbf{\Lambda}}_1^N$ (séquence de l'information à priori) devrait être indépendante des entrées du décodeur en cours conditionnellement sur les bits d'information. Par conséquent, nous devrons bien décomposer la sortie Λ_n en des parties afin de faire apparaître l'information à échanger entre les deux décodeurs. En utilisant les (3.1) et (3.3), nous pouvons exprimer Λ_n par:

$$\begin{aligned} \Lambda_n &= \ln \frac{Pr(u_n = 1|\mathbf{X}_1^N, \mathbf{Y}_1^N, \tilde{\mathbf{\Lambda}}_1^N)}{Pr(u_n = 0|\mathbf{X}_1^N, \mathbf{Y}_1^N, \tilde{\mathbf{\Lambda}}_1^N)} \\ &= \ln \frac{Pr(u_n = 1|\mathbf{X}_1^{n-1}, x_n, \mathbf{X}_{n+1}^N, \mathbf{Y}_1^N, \tilde{\mathbf{\Lambda}}_1^{n-1}, \tilde{\Lambda}_n, \tilde{\mathbf{\Lambda}}_{n+1}^N)}{Pr(u_n = 0|\mathbf{X}_1^{n-1}, x_n, \mathbf{X}_{n+1}^N, \mathbf{Y}_1^N, \tilde{\mathbf{\Lambda}}_1^{n-1}, \tilde{\Lambda}_n, \tilde{\mathbf{\Lambda}}_{n+1}^N)} \end{aligned} \tag{3.5}$$

Posons $\mathbf{V}_n = (\mathbf{X}_1^{n-1}, \mathbf{X}_{n+1}^N, \mathbf{Y}_1^N, \tilde{\mathbf{\Lambda}}_1^{n-1}, \tilde{\mathbf{\Lambda}}_{n+1}^N)$ avec $\mathbf{X}_{n+1}^N = (x_{n+1}, x_{n+2}, \ldots, x_N)$ et $\tilde{\mathbf{\Lambda}}_{n+1}^N = (\tilde{\Lambda}_{n+1}, \tilde{\Lambda}_{n+2}, \ldots, \tilde{\Lambda}_N)$.

Supposons que les variables x_n, $\tilde{\Lambda}_n$ et $\mathbf{V}_n$ sont indépendantes entre elles conditionnellement sur u_n alors Λ_n peut s'exprimer par:

$$\begin{aligned}
\Lambda_n &= \ln \frac{Pr(u_n = 1|x_n, \tilde{\Lambda}_n, \mathbf{V}_n)}{Pr(u_n = 0|x_n, \tilde{\Lambda}_n, \mathbf{V}_n)} \\
&= \ln \frac{Pr(x_n|u_n = 1)Pr(\tilde{\Lambda}_n|u_n = 1)Pr(\mathbf{V}_n|u_n = 1)Pr(u_n = 1)}{Pr(x_n|u_n = 0)Pr(\tilde{\Lambda}_n|u_n = 0)Pr(\mathbf{V}_n|u_n = 0)Pr(u_n = 0)} \\
&= \ln \frac{Pr(x_n|u_n = 1)Pr(u_n = 1|\tilde{\Lambda}_n)Pr(\mathbf{V}_n|u_n = 1)Pr(\tilde{\Lambda}_n)}{Pr(x_n|u_n = 0)Pr(u_n = 0|\tilde{\Lambda}_n)Pr(\mathbf{V}_n|u_n = 0)Pr(\tilde{\Lambda}_n)} \\
&= \ln \frac{Pr(x_n|u_n = 1)}{Pr(x_n|u_n = 0)} + \ln \frac{Pr(u_n = 1|\tilde{\Lambda}_n)}{Pr(u_n = 0|\tilde{\Lambda}_n)} + \ln \frac{Pr(\mathbf{V}_n|u_n = 1)}{Pr(\mathbf{V}_n|u_n = 0)} \qquad (3.6)
\end{aligned}$$

Or $Pr(\tilde{\Lambda}_n)$ est un facteur constant et commun aussi bien pour le numérateur que le dénominateur. De plus, dans le cas d'un canal Gausien, x_n est une variable aléatoire Gaussienne de moyenne $(2u_n - 1)$ et de variance σ^2. Nous avons alors:

$$p(x_n|u_n = i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_n - (2i-1))^2}{2\sigma^2}} \qquad (3.7)$$

D'autre part, nous savons que $Pr(u_n = 1) = 1 - Pr(u_n = 0)$ et en utilisant (3.2), les valeurs $Pr(u_n = 1|\tilde{\Lambda}_n)$ et $Pr(u_n = 0|\tilde{\Lambda}_n)$ peuvent s'exprimer par:

$$\begin{aligned}
Pr(u_n = 1|\tilde{\Lambda}_n) &= \frac{e^{\tilde{\Lambda}_n}}{1 + e^{\tilde{\Lambda}_n}} \qquad \text{et} \\
Pr(u_n = 0|\tilde{\Lambda}_n) &= \frac{1}{1 + e^{\tilde{\Lambda}_n}} \qquad (3.8)
\end{aligned}$$

En combinant (3.6), (3.7) et (3.8), Λ_n peut s'écrire comme:

$$\begin{aligned}
\Lambda_n &= \frac{2}{\sigma^2} x_n + \tilde{\Lambda}_n + Le_n \\
&= Li_n + La_n + Le_n \qquad (3.9)
\end{aligned}$$

Ainsi, le rapport de vraisemblance Λ_n est composé de trois termes. Le premier terme Li_n est appelé information intrinsèque puisqu'il contient la contribution directe de l'information systématique. Le deuxième terme détient la contribution de

l'information à priori $\tilde{\Lambda}_n$, il est désigné par La_n. Quant au dernier terme, il est appelé information extrinsèque. Cette appellation provient du fait que Le_n contribue dans le calcul de Λ_n en utilisant d'autres informations et non les variables $\tilde{\Lambda}_n$ et x_n. Elle peut alors améliorer la fiabilité du symbole x_n.

En décodage itératif, la séquence d'information intrinsèque est toujours disponible pour les deux décodeurs. Par conséquent, la seule partie du rapport de vraisemblance Λ_n à faire passer d'un décodeur à un autre est la séquence d'information extrinsèque. Ainsi, après avoir été entrelacée ou délacée, cette séquence sera transformée en une séquence d'information à priori. Autrement dit, au cours d'une itération, l'information extrinsèque obtenue à la sortie d'un décodeur devient une information à priori pour l'autre décodeur.

Finalement, Λ_n peut s'exprimer par :

$$\Lambda_n^k = Li_n + Le_{n'}^{k-1} + Le_n^k \tag{3.10}$$

où k désigne le numéro de l'itération, $Le_{n'}^{k-1}$ est l'information extrinsèque réordonnée générée par le précédent décodeur APP, et n' est l'indice indiquant l'effet de l'entrelaceur ou désentrelaceur.

La figure 3.2 illustre un décodeur APP avec ces différentes entrées-sorties. Dans la section suivante nous parlerons des algorithmes de décodage, utilisés par un décodeur APP.

Figure 3.2: Entrées-Sorties d'un décodeur APP

3.3 Algorithmes de décodage

En 1974, Bahl *et al* [1] ont présenté un algorithme de décodage optimal pour les codes convolutionnels désigné dans la littérature par l'algorithme de BCJR (*Bahl, Cocke, Jelinek, et Raviv*). L'appellation algorithme optimal provient du fait qu'il minimise le taux d'erreur par symbole et non par séquence de symboles.

Par rapport à l'algorithme de Viterbi qui consiste à minimiser la probabilité d'erreur d'une séquence de symboles, la complexité de l'algorithme de BCJR est très élevée. Par conséquent, il a été ignoré pendant longtemps.

En 1993, l'algorithme de BCJR a été modifié par Berrou *et al.* On l'appela alors algorithme MAP (Maximum A Posteriori). D'autres représentations de ce dernier, exprimées dans le domaine logarithmique, ont été proposées telle que Log-MAP et Max-Log-MAP que nous allons détailler ultérieurement.

Il existe un autre algorithme utilisé par le décodeur APP qui est une version modifiée de l'algorithme de Viterbi. En effet, dans [25] Hagenauer et Hoeher ont présenté une méthode qui estime la fiabilité d'un symbole décodé en utilisant l'algorithme de Viterbi. La version qui en a résulté est appelée SOVA (*Soft Output Viterbi Algorithm*). Par la suite des améliorations ont été faites pour l'algorithme SOVA. Résultat, dans [23] Fossorier *et al* ont montré une équivalence entre l'algorithme Max-Log-MAP et celui de SOVA.

Dans notre recherche, nous nous sommes intéressés seulement à l'algorithme MAP et à ses différentes variantes.

3.3.1 Algorithme MAP

Nous allons présenter brièvement l'algorithme MAP. Pour plus de renseignements, le lecteur est renvoyé à l'annexe II.

Comme nous l'avons vu dans le paragraphe précédent, les entrées du décodeur APP sont : ($\mathbf{X}_1^N$, $\mathbf{Y}_1^N$ et $\tilde{\mathbf{\Lambda}}_1^N$). Elles sont représentées par le vecteur $\mathbf{R}_1^N$. Considérant alors, $\mathbf{R}_n = (x_n, y_n, \tilde{\Lambda}_n)$ comme le vecteur qui se présente à l'entrée du décodeur à l'instant n. Ainsi, tel que montré à l'annexe II, la sortie pondérée de décodeur APP utilisant l'algorithme MAP peut être exprimée par:

$$\Lambda_n = \ln \frac{Pr(u_n = 1|\mathbf{R}_1^N)}{Pr(u_n = 0|\mathbf{R}_1^N)} = \ln \frac{\sum_{s'} \sum_s \alpha_{n-1}(s')\gamma_n^1(\mathbf{R}_n, s', s)\beta_n(s)}{\sum_{s'} \sum_s \alpha_{n-1}(s')\gamma_n^0(\mathbf{R}_n, s', s)\beta_n(s)} \tag{3.11}$$

où s' et s sont les états de codeur respectivement aux instants $n-1$ et n. Les quantités α, γ et β sont des probabilités définies dans l'annexe II. Elles sont appelées respectivement métrique d'état en avant, métrique de branche et métrique d'état en arrière. Elles sont reliées à la probabilité de transition des états du codeur et à la probabilité de transition du canal.

La métrique de branche $\gamma_n^i(\mathbf{R}_n, s', s)$, $i \in \{0, 1\}$ peut s'écrire:

$$\gamma_n^i(\mathbf{R}_n, s', s) = e^{\frac{(x_n(2i-1)+y_n(2c_n-1))}{\sigma^2} + iLa_n} . Pr(S_n = s|u_n = i, S_{n-1} = s') \tag{3.12}$$

où c_n est le bit de parité associé à la transition entre l'état s' et l'état s par l'introduction du symbole d'information u_n. La probabilité $Pr(S_n = s|u_n = i, S_{n-1} = s')$ est égale à 0 ou à 1. Elle dépend s'il y a ou non une transition dans le treillis entre l'état s' et l'état s.

Les valeurs de α et β peuvent être calculées par récurrence en commençant respectivement par le début et la fin du bloc reçu:

$$\alpha_n(s) = \sum_{s'=0}^{2^M-1} \sum_{i=0}^{1} \alpha_{n-1}(s')\gamma_n^i(\mathbf{R}_n, s', s) \tag{3.13}$$

$$\beta_n(s) = \sum_{s'=0}^{2^M-1} \sum_{i=0}^{1} \beta_{n+1}(s)\gamma_{n+1}^{i}(\mathbf{R}_{n+1}, s', s) \tag{3.14}$$

où M est la mémoire du codeur CRS.

Pour la mise en oeuvre de l'algorithme MAP, il est nécessaire d'initialiser les valeurs de récurrences α et β. Dans le cas où l'état initial et l'état final de deux codeurs sont zéro, on a:

$$\alpha_0(s) = \begin{cases} 1 & \text{si } s = 0 \\ 0 & \text{autrement} \end{cases} \tag{3.15}$$

et

$$\beta_N(s) = \begin{cases} 1 & \text{si } s = 0 \\ 0 & \text{autrement} \end{cases} \tag{3.16}$$

Mais si la terminaison vers l'état zéro est ignorée pour les deux codeurs alors l'initialisation de β deviendra :

$$\beta_N(s) = \frac{1}{2^M}, \qquad s = 0, 1, \ldots, 2^M - 1. \tag{3.17}$$

3.3.2 Algorithme Log-MAP

Le décodage utilisant l'algorithme MAP, nécessite beaucoup de mémoire et un grand nombre d'opérations (multiplications et exponentielles). Par conséquent, l'implémentation d'un tel algorithme dans un système de communication est une tâche très complexe.

Toutefois, l'utilisation de la représentation logarithmique des différentes quantités γ, α et β, notées respectivement par $\underline{\gamma}$, $\underline{\alpha}$ et $\underline{\beta}$, simplifie beaucoup la tâche de l'implémentation et diminue ainsi la complexité de l'algorithme MAP. La version résultante est appelée algorithme Log-MAP [32].

Les expressions de $\underline{\gamma}$, $\underline{\alpha}$ et $\underline{\beta}$ sont:

$$\underline{\gamma}_n^i(R_n, s', s) = \ln \gamma_n^i(\mathbf{R}_n, s', s) \tag{3.18}$$

$$\underline{\alpha}_n(s) = \ln \alpha_n(s) = \ln \sum_{s'=0}^{2^M-1} \sum_{i=0}^{1} e^{(\underline{\alpha}_{n-1}(s') + \underline{\gamma}_n^i(\mathbf{R}_n, s', s))} \tag{3.19}$$

$$\underline{\beta}_n(s') = \ln \beta_n(s') = \ln \sum_{s'=0}^{2^M-1} \sum_{i=0}^{1} e^{(\underline{\beta}_{n+1}(s) + \underline{\gamma}_{n+1}^i(\mathbf{R}_{n+1}, s', s))} \tag{3.20}$$

En introduisant ces nouvelles expressions dans (3.11), Λ_n peut s'écrire sous la forme suivante:

$$\begin{aligned}
\Lambda_n &= \ln \frac{\sum_{s'} \sum_{s} \alpha_{n-1}(s') \gamma_n^1(\mathbf{R}_n, s', s) \beta_n(s)}{\sum_{s'} \sum_{s} \alpha_{n-1}(s') \gamma_n^0(\mathbf{R}_n, s', s) \beta_n(s)} \\
&= \ln \frac{\sum_{s'} \sum_{s} e^{\underline{\alpha}_{n-1}(s')} e^{\underline{\gamma}_n^1(\mathbf{R}_n, s', s)} e^{\underline{\beta}_n(s)}}{\sum_{s'} \sum_{s} e^{\underline{\alpha}_{n-1}(s')} e^{\underline{\gamma}_n^0(\mathbf{R}_n, s', s)} e^{\underline{\beta}_n(s)}} \\
&= \ln \frac{\sum_{s'} \sum_{s} e^{(\underline{\alpha}_{n-1}(s') + \underline{\gamma}_n^1(\mathbf{R}_n, s', s) + \underline{\beta}_n(s))}}{\sum_{s'} \sum_{s} e^{(\underline{\alpha}_{n-1}(s') + \underline{\gamma}_n^0(\mathbf{R}_n, s', s) + \underline{\beta}_n(s))}}
\end{aligned} \tag{3.21}$$

D'après l'algorithme Jacobien [19], définit par la fonction Ω, l'expression $1 + a^{\psi}$ peut être réécrite par:

$$1 + a^{\psi} = a^{\Omega(\psi)}$$

Par analogie:

$$\begin{aligned}
e^{\psi_1} + e^{\psi_2} &= e^{\psi_1}(1 + e^{\psi_2 - \psi_1}) \\
&= e^{\psi_2}(1 + e^{\psi_1 - \psi_2}) \\
&= e^{\max(\psi_1, \psi_2) + \ln(1 + e^{-|\psi_1 - \psi_2|})}
\end{aligned} \tag{3.22}$$

Ou encore

$$\ln(e^{\psi_1} + e^{\psi_2}) = \max(\psi_1, \psi_2) + \ln(1 + e^{-|\psi_1 - \psi_2|}) \tag{3.23}$$

et d'une façon générale et récursive, on trouve:

$$\begin{aligned}\ln(e^{\psi_1}+e^{\psi_2}+\ldots+e^{\psi_n}) &= \ln(\Delta+e^{\psi_n}), \qquad \Delta = e^{\psi_1}+e^{\psi_2}+\ldots+e^{\psi_{n-1}} = e^{\psi} \\ &= \max(\ln\Delta, \psi_n) + \ln(1+e^{-|\ln\Delta-\psi_n|}) \\ &= \max(\psi, \psi_n) + \ln(1+e^{-|\psi-\psi_n|}) \qquad (3.24)\end{aligned}$$

Une conséquence de (3.24) est que les expressions de $\underline{\gamma}$, $\underline{\alpha}$, $\underline{\beta}$ et Λ_n peuvent être encore simplifiées et rendant ainsi l'algorithme Log-MAP moins complexe. Toutefois, les performances de cet algorithme sont les mêmes que celle de MAP.

Avec l'algorithme Log-MAP, les initiations des différentes quantités $\underline{\alpha}$ et $\underline{\beta}$ sont:

- Si les deux codeurs commencent et finissent à l'état zéro alors:

$$\alpha_0(s) = \begin{cases} 0 & \text{si } s=0 \\ -\infty & \text{autrement} \end{cases} \qquad \text{et } \beta_N(s) = \begin{cases} 0 & \text{si } s=0 \\ -\infty & \text{autrement} \end{cases} \qquad (3.25)$$

- si l'état final des deux codeurs est inconnu alors:

$$\begin{aligned}\alpha_0(s) &= \begin{cases} 0 & \text{si } s=0 \\ -\infty & \text{autrement} \end{cases} \\ \text{et } \beta_N(s) &= -M\ln 2, \qquad s=0,1,\ldots,2^M-1. \qquad (3.26)\end{aligned}$$

3.3.3 Algorithme Max-Log-MAP

Afin de réduire la complexité de l'algorithme Log-MAP, l'expression de (3.24) peut être simplifiée par l'approximation suivante:

$$\ln(e^{\psi_1}+e^{\psi_2}+\ldots+e^{\psi_n}) \approx \max_{i\in\{1,2,\ldots,n\}} \psi_i \qquad (3.27)$$

Ainsi toutes les opérations de multiplication dans l'algorithme MAP sont remplacées par des additions. La version obtenue est désignée par algorithme MAP

sous optimal ou bien Max-Log-MAP [38]. Elle est identique de point de vue complexité et performance à celle de SOVA.

3.4 Décodage itératif Turbo

Le décodage turbo s'effectue en mode série comme l'indique le schéma de la figure 3.3.

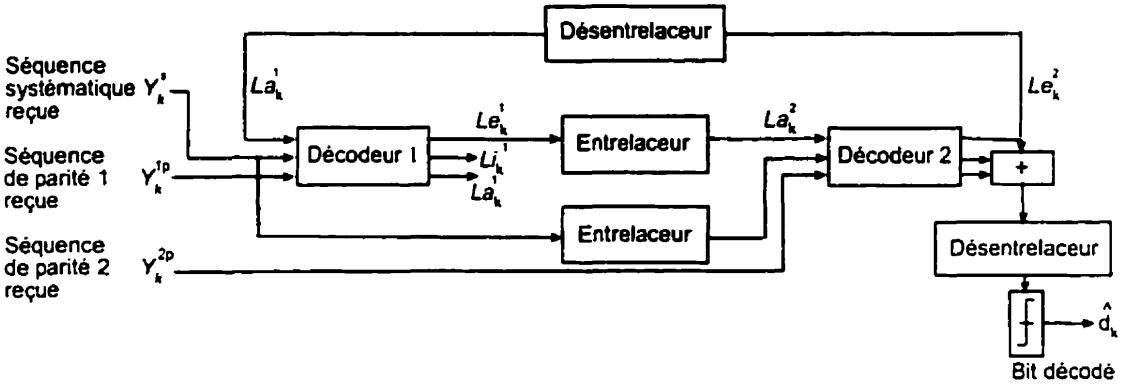


Figure 3.3: Décodeur turbo

Dans la suite, nous appelons DEC1 le décodeur avec l'entrée de parité 1 et DEC2 le décodeur avec l'entrée de parité 2.

Chaque décodeur génère, pour chaque symbole, une valeur appelée Probabilité à Posteriori (APP) en utilisant la sortie de l'autre décodeur comme entrée à priori [8]. Cette procédure se répète jusqu'à atteindre une condition prédéterminée qui représente le critère d'arrêt de la procédure.

A la $i^{\text{ème}}$ itération, le DEC1 reçoit l'information systématique Y_k^s, la parité 1 (Y_k^{1p}) et l'information à priori $La_k^1(i) = Le_k^{f2}(i-1)$ (qui est égale à zéro au début de décodage : 1ère itération) pour générer les informations extrinsèques $\mathbf{Le}^1(i)$ qui seront entrelacées avant d'être transmises au DEC2 comme entrée à priori $\mathbf{La}^2(i)$.

A l'instant k, le LLR à la sortie du DEC1 est égal à:

$$\Lambda_k^1(i) = \frac{2}{\sigma^2} Y_k^s + La_k^1(i) + Le_k^1(i) \tag{3.28}$$

Le DEC2 dispose de $Le_k'^1(i)$ (information extrinsèque entrelacée), de l'information systématique entrelacée $Y_k'^s$ et de la parité 2 (Y_k^{2p}) pour générer l'information extrinsèque $Le_k^2(i)$ qui sera délacée à son tour, puis retournée, via la boucle de retour, vers le DEC1. Le LLR à la sortie du DEC2 est égal à:

$$\Lambda_k^2(i) = \frac{2}{\sigma^2} Y_k'^s + La_k^2(i) + Le_k^2(i) \tag{3.29}$$

et ainsi la performance de décodage s'améliorera d'une itération à une autre jusqu'à la saturation. A ce moment, la valeur de la probabilité à posteriori sera quantifiée pour générer la séquence des bits décodés qui sera délacée c'est-à-dire remise dans l'ordre original.

Dans l'annexe II et plus particulièrement avant le développement de (II.16), l'indépendance entre les entrées d'un décodeur à l'instant k conditionnellement sur la transition de l'état S_{k-1} vers S_k doit être assurée. Sinon, l'algorithme MAP ne peut pas être utilisé. Or, dans le cas du code turbo et durant la première itération cette condition est encore valide, mais à partir de la deuxième, une certaine dépendance peut exister.

En effet, au début de décodage l'information à priori du décodeur 1 est égale à zéro donc il n'y a pas de dépendance entre les entrées de ce dernier. Pour le deuxième décodeur l'information à priori est indépendante des autres entrées puisqu'elles n'ont pas contribué à sa génération. Cependant, à partir de la deuxième itération et à l'instant k, le décodeur 1 reçoit une information à priori dépendante de Y_k^s et de Y_k^{1p}. Par conséquent, il y aura une violation de la condition d'utilisation de l'algorithme MAP.

Toutefois, la présence des entrelaceurs diminuent la corrélation entre les entrées de décodeurs. Plus la distance entre les symboles initialement proches est grande après l'entrelacement plus la décorrélation est élevée. En effet, dans leurs positions initiales i et j, les symboles occupent après entrelacement k et l comme nouvelles positions: plus $|k-l|$ est supérieur à $|i-j|$, meilleures sont les performances.

La saturation du code turbo provient de mauvais choix d'une part du type et de la taille des entrelaceurs et d'autre part des codeurs élémentaires CRS qui contribuent peu à augmenter la distance libre d_{free} de l'encodeur turbo [31]. Plus les tailles des entrelaceurs sont élevées, moins les entrées d'un tel décodeur sont corrélées.

3.5 Résultats de simulation

Le code turbo est le seul type de code correcteur d'erreurs qui se rapproche assez près de la limite de Shannon. Cependant, son comportement dépend de beaucoup de paramètres. Dans cette section, nous allons présenter les effets de la taille de l'entrelaceur, l'influence de la longueur de mémoire des codes CRS et aussi celle du nombre de ses boucles de retour sur la performance d'erreur des codes turbo. L'effet de choix du canal et du nombre d'itérations vont être également évoqués.

Tous les résultats de simulation présentés sont basés sur le décodage itératif turbo en utilisant l'algorithme Log-MAP, sauf à la dernière section où une comparaison de performance sera faite entre cet algorithme et celui de Max-Log-MAP.

3.5.1 Effet de la taille de l'entrelaceur sur la performance des codes turbo

La longueur des entrelaceurs est un paramètre très critique pour la performance des codes turbo. Elle affecte les propriétés de distance, en particulier la distance

libre du code utilisé. L'augmentation de la taille des entrelaceurs produira un effet de désordre plus élevé, donc une diminution de la corrélation entre les entrées de décodeurs.

La figure 3.4, montre bien qu'une augmentation de la taille des entrelaceurs engendre une amélioration de la performance. Cependant et au-delà d'une certaine valeur de cette taille, il serait inutile d'augmenter davantage car l'amélioration apportée ne fournit pas beaucoup de gain mais plutôt un accroissement du délai surtout pour des rapports signal sur bruit élevés.

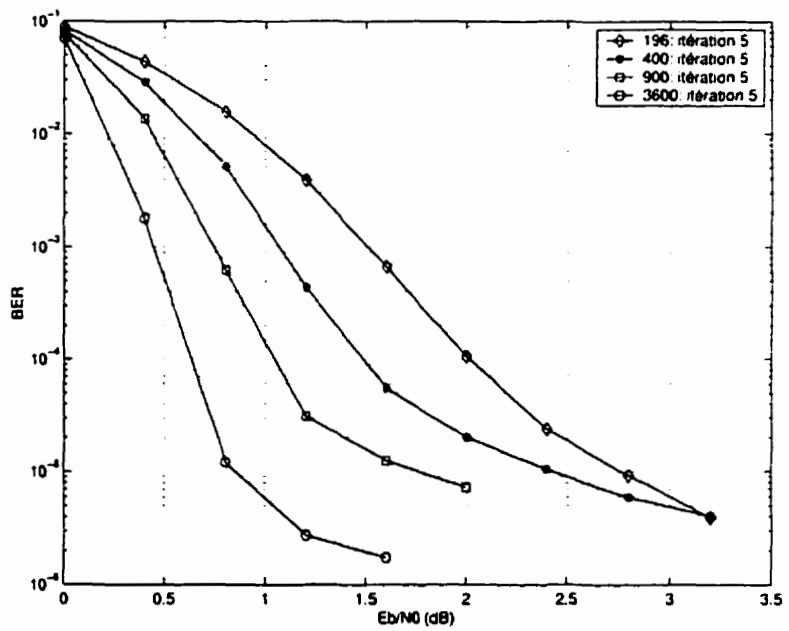


Figure 3.4: Influence de la taille de l'entrelaceur aléatoire sur la performance des codes turbo dans un canal AWGN avec $K = 5$, $G = (1, 21/37)$ et $R_{gp} = 1/3$

Nous remarquerons que plus la taille de l'entrelaceur est faible, plus vite le code turbo sature. Cela est dû à une diminution de la distance libre qui est proportionnelle aussi bien à la taille que le type des entrelaceurs [6].

Pour des faibles rapports signal sur bruit, la taille des entrelaceurs joue un rôle très important pour diminuer la probabilité d'erreur. Suivant le type des entrelaceurs, plus la taille est élevée plus le code turbo se rapproche de la limite de Shannon [8].

3.5.2 Effet du type de l'entrelaceur sur la performance des codes turbo

Dans le chapitre 2, nous avons classé les entrelaceurs en trois grandes classes: les entrelaceurs convolutionnels, les entrelaceurs en blocs et les entrelaceurs aléatoires. Pour connaître les effets des entrelaceurs sur la performance des codes turbo, nous allons procéder à une étude par classe et par la suite sélectionner et comparer les meilleurs d'entre eux.

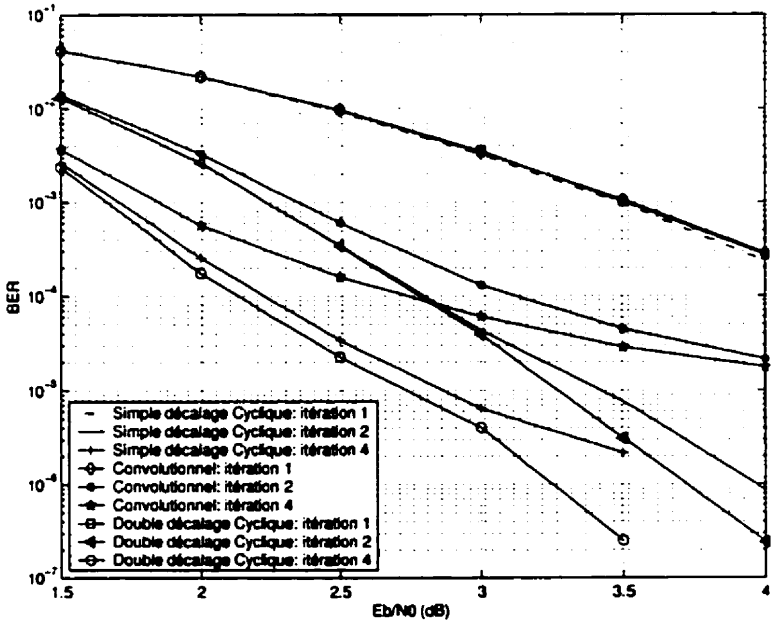


Figure 3.5: Performance des codes turbo utilisant différents entrelaceurs convolutionnels de taille 900 dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$ et $R_{gp} = 1/2$

- ***Classe des entrelaceurs convolutionnels:*** suivant la figure 3.5 le meilleur entrelaceur est l'entrelaceur à double décalage cyclique. Cet entrelaceur présente aussi une plus grande distance libre que les autres types puisque la convergence (saturation) de l'algorithme MAP arrive un peu en retard.

- ***Classe des entrelaceurs en bloc:*** d'après la figure 3.6 comme première remarque, nous pouvons affirmer que l'entrelaceur hélicoïdal est meilleur que l'entrelaceur bloc classique. En deuxième constation et comme il est dit dans le chapitre 2, les résultats de simulation prouvent que l'entrelaceur hélicoïdal BH de la figure 2.9 fournit plus de gain de codage que celui HB de la figure 2.8.

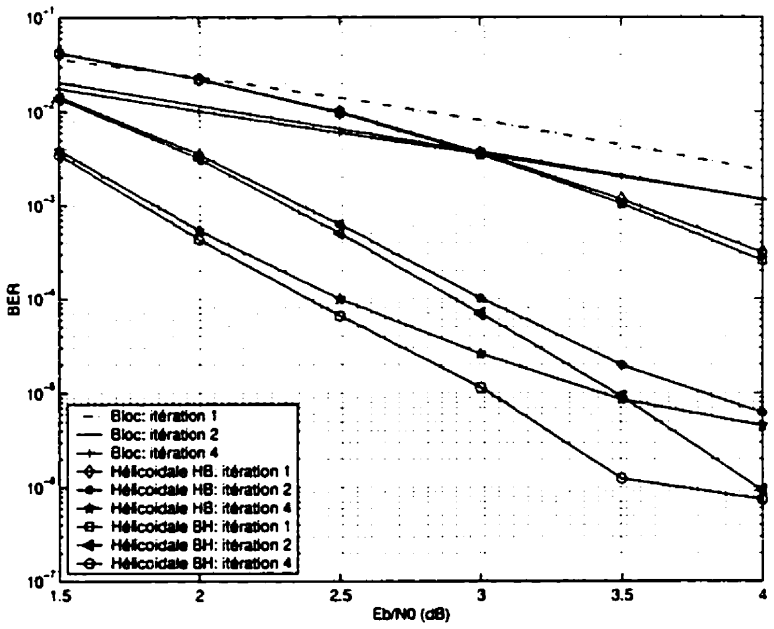


Figure 3.6: Performance des codes turbo utilisant différents entrelaceurs blocs de taille 900 dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$ et $R_{gp} = 1/2$

- ***Classe des entrelaceurs aléatoires:*** le résultat de simulation de la figure 3.7, montre clairement que l'entrelaceur S-aléatoire fournira les meilleures

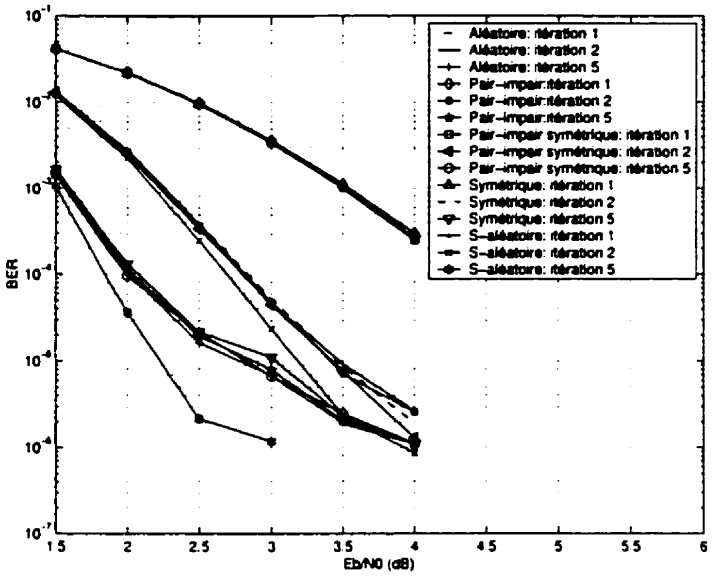


Figure 3.7: Performance des codes turbo utilisant différents entrelaceurs aléatoires de taille 900 dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$ et $R_{gp} = 1/2$

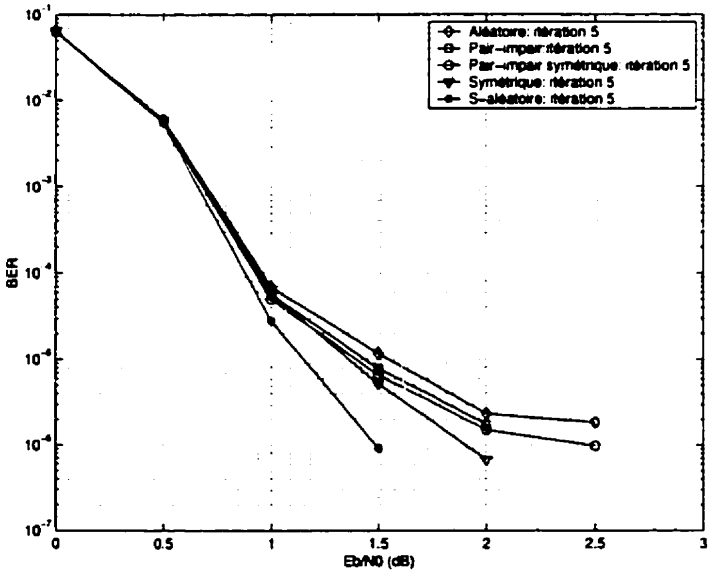


Figure 3.8: Performance des codes turbo utilisant différents entrelaceurs aléatoires de taille 4096 dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$ et $R_{gp} = 1/2$

performances pour le code turbo. Cependant, il est très complexe à mettre en oeuvre et lors de sa génération il présente plus de retard que les autres entrelaceurs aléatoires [39].

Pour montrer la différence entre les autres types d'entrelaceur aléatoires, nous avons augmenté la taille des différents entrelaceurs afin de pouvoir comparer l'influence apportée par chacun entre eux. A partir de la figure 3.8, nous pouvons classer ces différents entrelaceurs du plus mauvais au meilleur comme suit: pseudo-aléatoire, pseudo-aléatoire pair-impair, symétrique, S-aléatoire.

3.5.3 Effet du choix des codes CRS sur la performance des codes turbo

Généralement, une augmentation de la mémoire des codes élémentaires contribue à une augmentation de la distance libre donc à une amélioration des performances asymptotiques des codes turbo.

Avant d'exposer l'influence de la mémoire des codes CRS sur la performance des codes turbo, il serait bien de montrer le comportement de ces derniers pour une même longueur de contrainte mais avec différents vecteurs générateurs $G = (1, G2/G1)$. L'idée est de verifier l'influence du nombre de boucles de retour : la valeur du dénominateur $G1$ sur la performance des codes turbo. A priori, nous avons pensé que si la valeur de $G1$ augmente alors il y aura sûrement plus d'amélioration. Ceci est vrai seulement pour des faibles rapports signal sur bruit, comme l'indique la figure 3.9 et 3.10 résultats de simulation avec un entrelaceur aléatoire, taux de codage $R_{gp} = 1/3$ et pour des longueurs de contrainte respectivement $K = 5$ et $K = 4$. Pour des valeurs de $E_b/N_0 > 0.5$ dB les codes CRS avec des faibles $G1$ apportent plus de gain de codage et moins de saturation.

Ce comportement reste le même pour des valeurs de taux de codage élevées.

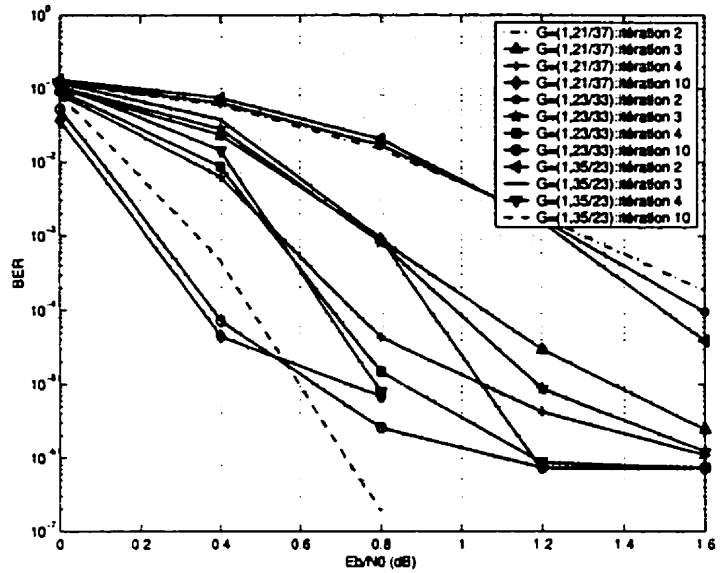


Figure 3.9: Influence du choix des codeurs CRS de longueur $K = 5$ sur la performance du code turbo dans un canal AWGN avec $R_{gp} = 1/3$ et taille de l'entrelaceur = 4096

Avec une mémoire très faible $M = 2$ nous remarquons d'après la figure 3.11 que le code turbo présente des meilleurs performances avec $G1$ élevé pour toutes les valeurs de E_b/N_0. Vu que les performances en terme de probabilité d'erreur pour les deux codeurs élémentaires sont très loins de 10^{-5}, la légère différence entre les deux courbes de cette figure est ignorée surtout à E_b/N_0 faible.

L'explication liée à ce phénomène reste encore inconnue. Nous avons même étudié le comportement de la distribution de poids pour les différents codes déjà mentionnés, mais aucune interprétation valide peut être déduite.

Nous nous intéressons maintenant à l'effet de la mémoire sur la performance du code turbo. D'après ce que nous avons dit précédemment, il faudrait bien choisir l'intervalle des valeurs de E_b/N_0 avec leurs meilleurs vecteurs générateurs pour chaque valeur de longueur de contrainte K. Les figures 3.12 et 3.13, résultats

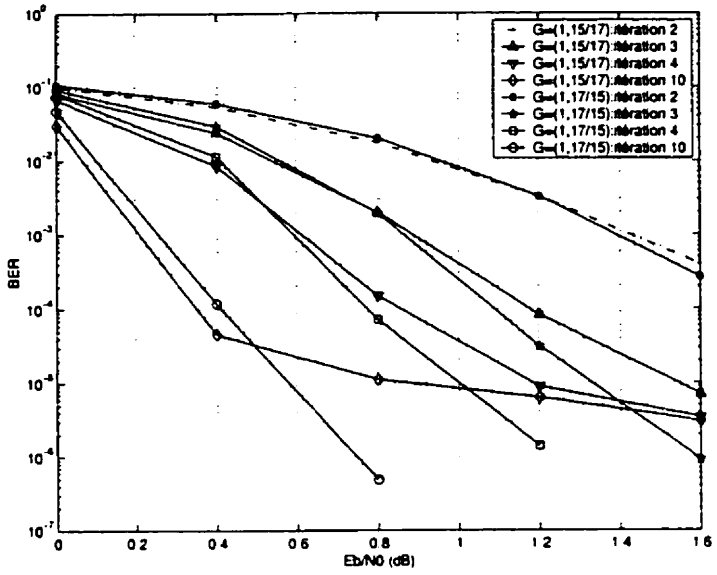


Figure 3.10: Influence du choix des codeurs CRS de longueur $K = 4$ sur la performance du code turbo dans un canal AWGN avec $R_{gp} = 1/3$ et taille de l'entrelaceur = 4096

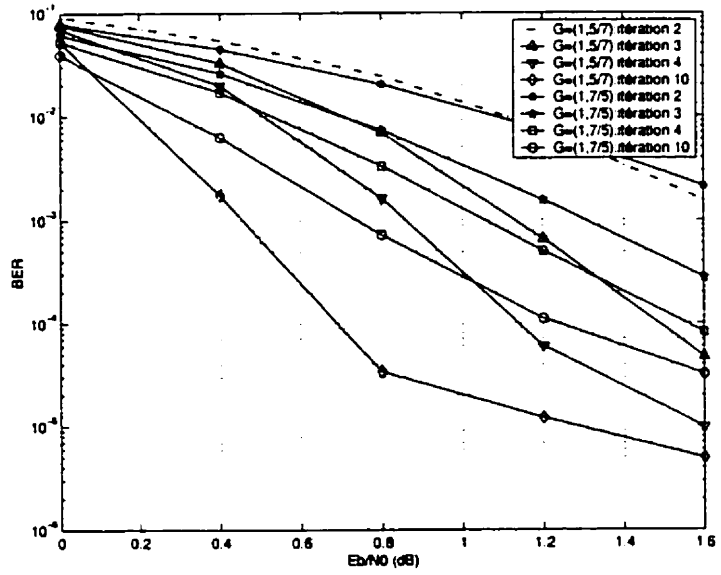


Figure 3.11: Influence du choix des codeurs CRS de longueur $K = 3$ sur la performance du code turbo dans un canal AWGN avec $R_{gp} = 1/3$ et taille de l'entrelaceur = 4096

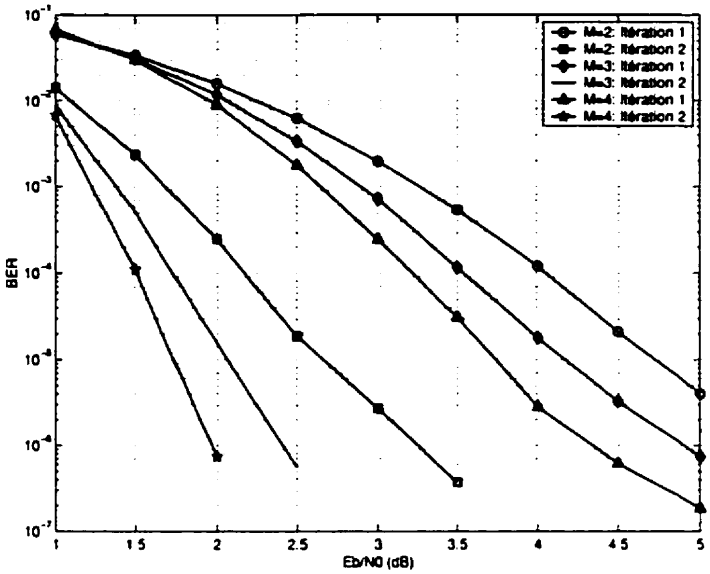


Figure 3.12: Influence de la longueur de la mémoire des codeurs CRS sur la performance du code turbo dans un canal AWGN avec $R_{gp} = 1/3$ et taille de l'entrelaceur aléatoire 900

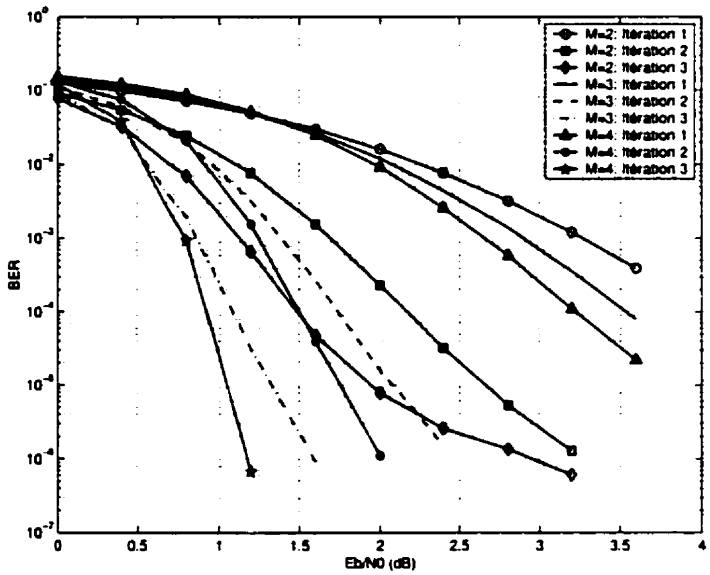


Figure 3.13: Influence de la longueur de la mémoire des codeurs CRS sur la performance du code turbo dans un canal AWGN avec $R_{gp} = 1/3$ et taille de l'entrelaceur aléatoire 4096

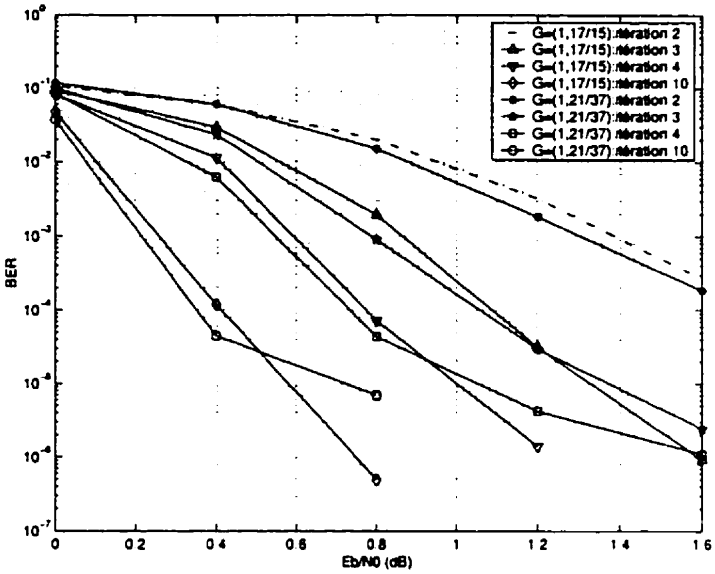


Figure 3.14: Influence du choix des codeurs CRS sur la performance du code turbo dans un canal AWGN avec $R_{gp} = 1/3$ et taille de l'entrelaceur aléatoire 4096

de simulation avec des tailles d'entrelaceur aléatoires respectivement 900 et 4096, montrent clairement qu'une augmentation de la longueur de mémoire engendre une meilleure performance mais en contre partie l'algorithme MAP deviendra plus complexe. D'autre part, la figure 3.14 illustre qu'il est inutile d'utiliser le vecteur $G = (1, 21/37)$ avec $K = 5$ dans le cas du code turbo de taux de codage $R_{gp} = 1/3$ avec $E_b/N_0 > 0.5$ dB, puisque les performances obtenues avec une longueur de contrainte $K = 4$ et $G = (1, 17/15)$ seront meilleures à partir d'une telle valeur.

3.5.4 Effet du nombre d'itérations sur la performance des codes turbo

Considérons un code turbo de taux de codage $R_{gp} = 1/3$ avec un vecteur générateur $G = (1, 21/37)$ et un entrelaceur aléatoire. Les résultats de simulation sont illustrés aux figures 3.15 et 3.16 correspondant respectivement aux tailles d'entrelaceur 22500 et 4096.

L'augmentation du nombre d'itérations joue un effet considérable sur la performance en terme de probabilité d'erreur. Cependant, il présentera une limite à partir de laquelle les mêmes performances seront obtenues selon la taille des entrelaceurs, la caractéristique des codes élémentaires et le rapport signal sur bruit. Par exemple, Avec l'entrelaceur de taille 4096 et au delà de 8 itérations, il n'y aucune amélioration par contre avec une taille de 22500 la saturation est atteinte à la $10^{ème}$ itération.

Lors de la saturation c'est à dire convergence de l'algorithme, il serait alors important d'interrompre le processus de décodage et limiter le nombre d'itérations par l'utilisation des critères d'arrêt. Différents critères seront présentés dans la prochaine section.

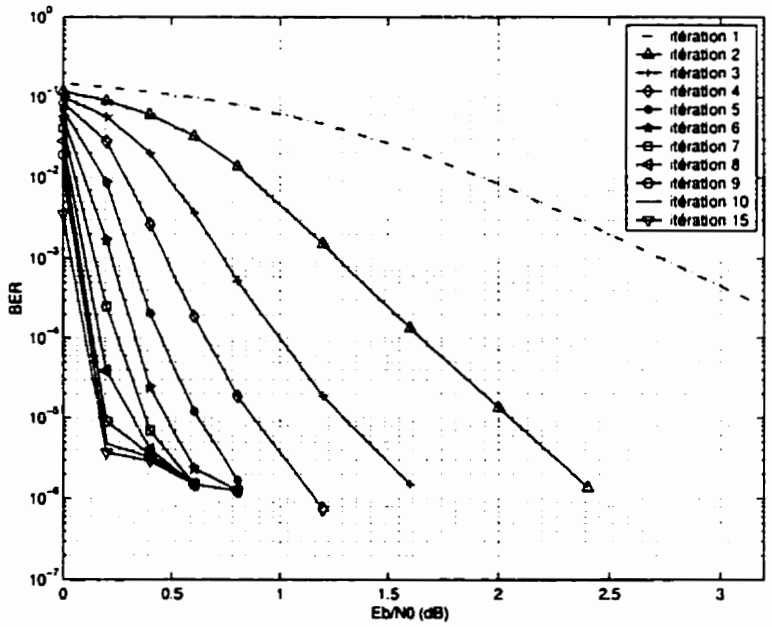


Figure 3.15: Influence du nombre d'itérations sur la performance des codes turbo dans un canal AWGN avec $K = 5$, $G = (1, 21/37)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 22500

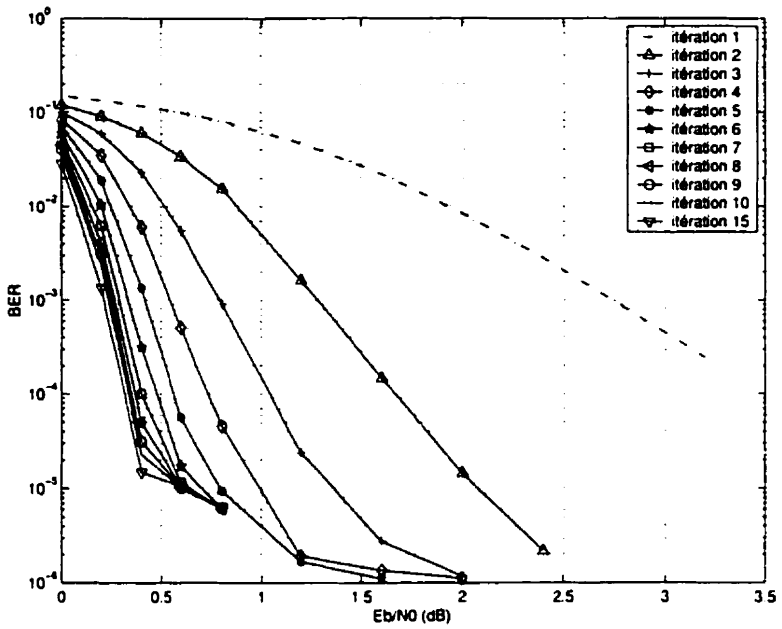


Figure 3.16: Influence du nombre d'itérations sur la performance des codes turbo dans un canal AWGN avec $K = 5$, $G = (1, 21/37)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 4096

3.5.5 Effet du choix du type de canal sur la performance des codes turbo

Les performances en terme de probabilité d'erreur par bit du code turbo dans les canaux de Rayleigh et Gaussien sont obtenues par simulations. Les résultats de ces simulations sont illustrés dans la figure 3.17 en utilisant un taux de codage $R_{gp} = 1/3$, une mémoire de valeur 4, taille d'entrelaceur aléatoire 900 et un vecteur générateur $G = (1, 21/37)$.

Pour la même valeur de rapport signal sur bruit et le même nombre d'itérations, nous remarquons que le rapport obtenu entre les probabilités d'erreurs avec le canal Gaussien et celui de Rayleigh est au moins 10^{-3}. Aussi, après deux itérations, nous pouvons dire qu'avec un canal de Rayleigh la performance chute de 1.5 dB par rapport à un canal Gaussien [30].

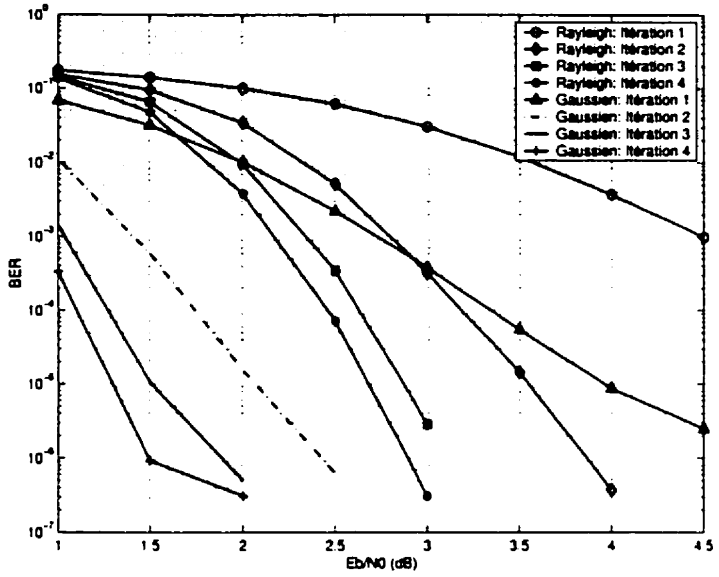


Figure 3.17: Performance du code turbo dans un canal AWGN et Rayleigh avec $K = 5$, $G = (1, 21/37)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 900

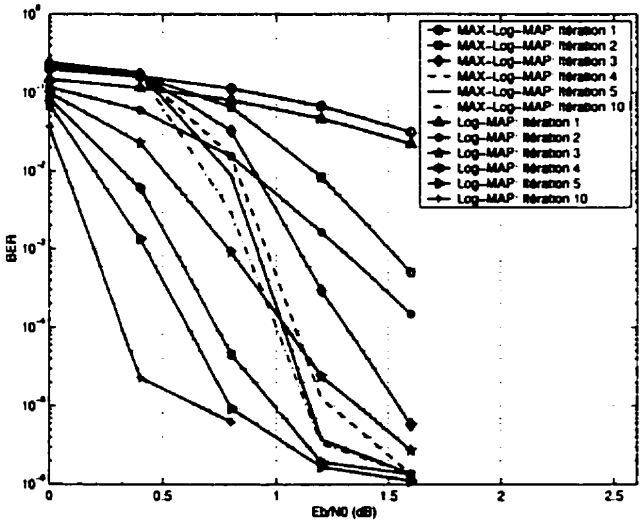


Figure 3.18: Performance du code turbo utilisant l'algorithme Log-MAP et Max-Log-MAP avec $K = 5$, $G = (1, 21/37)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 900

3.5.6 Performance des codes turbo utilisant l'algorithme Max-Log-MAP

La figure 3.18 montre les performances des codes turbo utilisant l'algorithme Log-MAP et Max-Log-MAP. Nous remarquons que les performances chutent surtout pour des faibles rapports signal sur bruit. Cependant, dans le cas d'un entrelaceur aléatoire de taille 900 et un codeur CRS de vecteur générateur $G = (1, 21/37)$ avec mémoire 4, l'algorithme Max-Log-MAP peut être utilisé sans aucune dégradation de la performance pour des valeurs de $E_b/N_0 > 1.2dB$.

3.6 Critères d'arrêt pour le décodage turbo

Le décodage turbo se fait d'une manière itérative. Le nombre d'itérations est limité puisque après un certain nombre, le décodage sera saturé et les performances ne s'améliore plus. Afin de contrôler et arrêter ce processus et gagner par la suite plus de temps d'exécution, une condition appelée critère d'arrêt devrait être vérifiée après chaque itération. Dans la littérature, nous trouvons plusieurs critères d'arrêt. Nous allons dans la suite les citer, les modifier et ainsi présenter notre solution.

Nous nous sommes basés sur les modifications introduites aux anciens critères et celui que nous proposons, pour faire le contrôle d'arrêt aussi bien à la sortie de DEC1 que le DEC2. Par conséquent, la décision sur les bits décodés se fait à la sortie des deux décodeurs comme l'indique la figure 3.19. Par cette modification, nous espérons avoir encore une diminution du délai de décodage au moins d'une moitié d'itération.

3.6.1 Entropie croisée (CE: Cross Entropy)

L'entropie croisée est utilisée pour mesurer la ressemblance entre deux distributions. Dans sa version initiale [26], ce critère est définit comme suit:

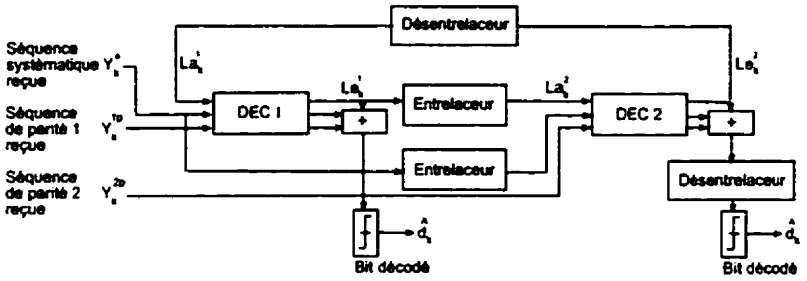


Figure 3.19: Schéma d'une double décision dans le décodage turbo

A l'instant k, la différence entre l'information extrinsèque à la $i^{\text{ème}}$ itération et celle de $(i-1)^{\text{ème}}$ pour le DEC2 est:

$$\triangle Le_k^2(i) = Le_k^2(i) - Le_k^2(i-1), i \geq 1 \tag{3.30}$$

Ainsi, l'entropie croisée $T_2(i)$ correspondant au DEC2 à la $i^{\text{ème}}$ itération, peut être évaluée approximativement par:

$$T_2(i) \approx \sum_{k=1}^{N} \frac{(\triangle Le_k^2(i))^2}{e^{|\Lambda_k^1(i)|}}, i \geq 1 \tag{3.31}$$

Finalement, le critère d'arrêt est le suivant:

$$\text{Si } T_2(i)/T_2(1) = 10^{-2} \sim 10^{-4}, \quad \text{on arrête le décodage}$$

Nous pouvons modifier et encore améliorer ce critère, c'est-à-dire diminuer encore le temps de décodage, par la détermination de l'entropie croisée aussi bien à la sortie du DEC1 que le DEC2. Nous désignons par CEM le critère d'arrêt résultant.

Pour ce faire, nous devons aussi déterminer la différence entre l'information extrinsèque à la $i^{\text{ème}}$ itération et celle de $(i-1)^{\text{ème}}$ pour le DEC1, qui est définie par:

$$\triangle Le_k^1(i) = Le_k^1(i) - Le_k^1(i-1), i > 1 \tag{3.32}$$

Et ainsi, l'entropie croisée $T_1(i)$ correspondant au DEC1 est :

$$T_1(i) \approx \sum_{k=1}^{N} \frac{(\triangle Le_k^1(i))^2}{e^{|\Lambda_k^2(i-1)|}}, i > 1 \tag{3.33}$$

Dans ce cas le critère d'arrêt serait d'une façon générale:

$$\text{Si } T_n(i)/T_2(1) = 10^{-2} \sim 10^{-4},\ (n = 1, 2) \text{ on arrête le décodage}$$

Cette modification a été faite au prix d'une augmentation de complexité. En fait, il est clair que le contrôle du critère d'arrêt à la sortie du premier et du deuxième décodeur demande deux fois plus d'opérations de calcul que s'il se fait seulement à la sortie du deuxième.

3.6.2 Critère d'arrêt SCR (Sign-Change-Ratio)

Soit $C^2(i)$ le nombre de changement de signe entre les éléments de la séquence d'information extrinsèque $\mathbf{Le}^2(i-1)$ à la $(i-1)^{\text{ème}}$ itération et ceux de la séquence d'information extrinsèque $\mathbf{Le}^2(i)$ à la $i^{\text{ème}}$ itération avec $i > 1$ [43].

Le critère SCR est très simple par rapport à celui de CE. Il s'agit de contrôler le rapport $\frac{C^2(i)}{N}$. S'il est inférieur à $\leq (0.005 \sim 0.03)$ alors le décodage peut être arrêté sans presque aucune dégradation de la performance.

Nous pouvons encore modifier ce critère par l'évaluation du nombre de changements de signe entre les éléments de la séquence d'information extrinsèque $\mathbf{Le}^1(i-1)$ et $\mathbf{Le}^1(i)$. Par conséquent, nous devons déterminer le rapport $\frac{C^n(i)}{N}$, $(n = 1, 2)$ qui doit être $\leq (0.005 \sim 0.03)$ pour arrêter le décodage. Nous désignons par SCRM le critère d'arrêt résultant. Dans ce cas, la complexité est le double de celle du SCR classique.

3.6.3 Critère d'arrêt HDA (Hard-Decision-Aided)

L'information de fiabilité à la fin de chaque itération peut fournir une information concernant la convergence du processus de décodage [43].

A la $(i-1)^{\text{ème}}$ itération, on sauvegarde la séquence d'information de fiabilité $\Lambda^2(i-1)$ dans une mémoire tampon. Lors de la $i^{\text{ème}}$ itération, on compare la séquence d'information de fiabilité $\Lambda^2(i)$ à celle de $\Lambda^2(i-1)$. Si les éléments de ces deux dernières séquences présentent les même signes, alors le décodage pourrait s'arrêter.

Nous pouvons appliquer la même procédure à la sortie du DEC1, c'est à dire comparer $\Lambda^1(i-1)$ et $\Lambda^1(i)$. Nous désignons par HDAM le critère d'arrêt résultant. Dans ce cas, il est clair que nous avons besoin de deux mémoires tampon, donc complexité double de celle du SCR.

3.6.4 Critère d'arrêt HDA-mixed

Dans HDA modifié (HDAM), nous comparons les séquences d'information de fiabilités $\Lambda^1(i)$ avec $\Lambda^1(i-1)$ ou bien $\Lambda^2(i)$ à $\Lambda^2(i-1)$. Il serait préférable de comparer à chaque demie d'itération les séquences $\Lambda^1(i)$ avec $\Lambda^2(i-1)$ ou bien $\Lambda^2(i)$ avec $\Lambda^1(i)$. Autrement dit, à chaque sortie de décodeur nous comparons les signes des éléments de la séquence d'information de fiabilité courante à la précédente sortie de fiabilité de l'autre décodeur. Dans ce cas, nous n'avons besoin que d'une seule mémoire tampon, pour garder une copie de la séquence de fiabilité de DEC1 ou DEC2. Cependant, à chaque itération cette mémoire est utilisée deux fois pour le sauvegarde des séquences.

3.6.5 Comparaisons de la performance des différents critères d'arrêt

L'efficacité des critères d'arrêt déjà mentionnés dépend de la taille de l'entrelaceur, du type de codeurs CRS et du rapport signal sur bruit E_b/N_0. En effet, pour des faibles rapports signal à bruit et pour des petites tailles d'entrelaceur, les décodeurs n'arrivent parfois pas à satisfaire la condition d'arrêt.

Pour résoudre ce problème, nous avons suggéré une condition de plus pour chaque critère:

- CE: Si le rapport $T_n(i)/T_2(1)$, $(n = 1, 2)$ reste constant après un certain nombre d'itérations i alors le décodage pourrait être arrêté.
- SCR: Si le rapport $\frac{C^n(i)}{N} = \frac{C^n(i-1)}{N}$, $(n = 1, 2)$ avec le nombre d'itérations i alors il serait inutile de continuer le décodage itératif.
- HDA: à chaque itération, si le nombre des signes des éléments de la séquence d'information de fiabilité courante égal le nombre des signes de ceux de la séquence de fiabilité précédente alors le décodage pourrait être arrêté.

A partir d'une petite étude comparative entre les figures 3.20, 3.21, 3.22, 3.23, 3.24, 3.25, 3.26 et 3.27, nous pouvons déduire que le meilleur critère d'arrêt est l'entropie croisée aussi bien pour la version initiale que celle modifiée. Cependant, vu la complexité supplémentaire introduite par ce dernier nous préférons les critères d'arrêt HDA, HDAM ou bien HDAMixed qui présentent une faible dégradation de la performance avec moins de complexité. Par ailleurs, les modifications du critère HDA (en HDAM et HDAMixed) ont permis de gagner plus de délai par l'interruption de processus de décodage turbo non seulement à la sortie du DEC2 mais aussi à celle de DEC1.

Tableau 3.1: Différents critères d'arrêt en mode série avec N est la longueur de la séquence d'information binaire

	CE	CEM	SCR	SCRM	HDA	HDAM	HDAMixed
opérations	5N-1	10N-2	3N-1	6N-2	N	2N	2N
mémoires	N+2	2N+4	N+2	2N+4	N	2N	N

3.7 Conclusion

Dans ce chapitre, nous avons décrit le principe de décodage itératif et son application dans le cas du code turbo. L'atout de cette technique est l'information extrinsèque qui peut améliorer les performances d'un tel décodeur sous certaines conditions. L'algorithme MAP et ses différentes variantes ont été abordées. Une étude des performances du code turbo suivant différents paramètres a été aussi exposée.

Par ailleurs, nous avons décrit différents critères d'arrêt pour pouvoir interrompre efficacement le décodage du code turbo. Dans le prochain chapitre, nous allons présenter le principe de décodage parallèle des codes turbo.

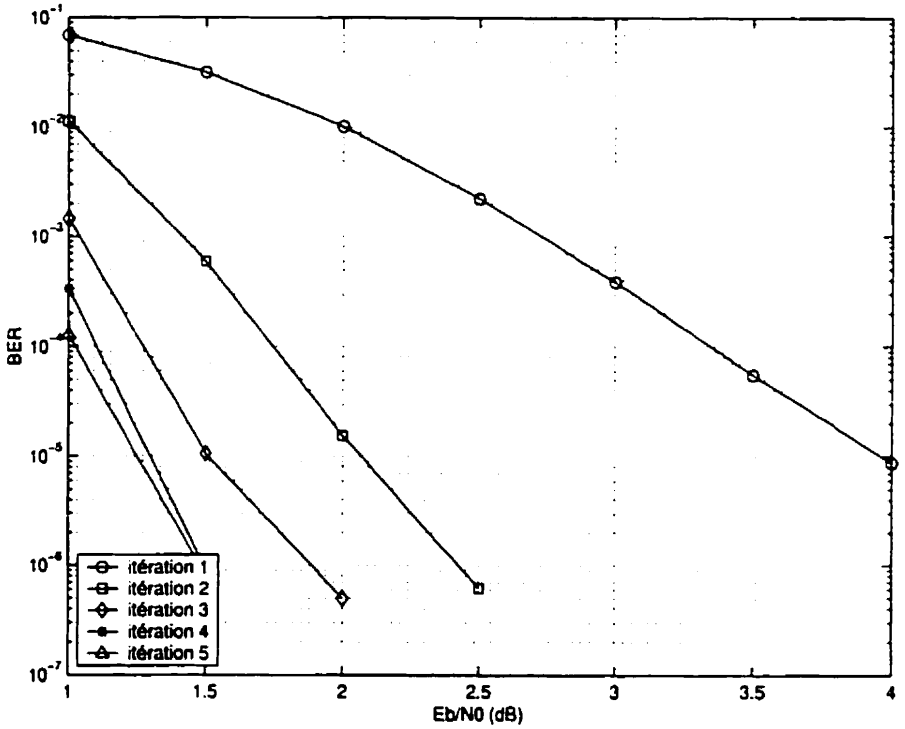


Figure 3.20: Performance du code turbo sans critère d'arrêt dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 900

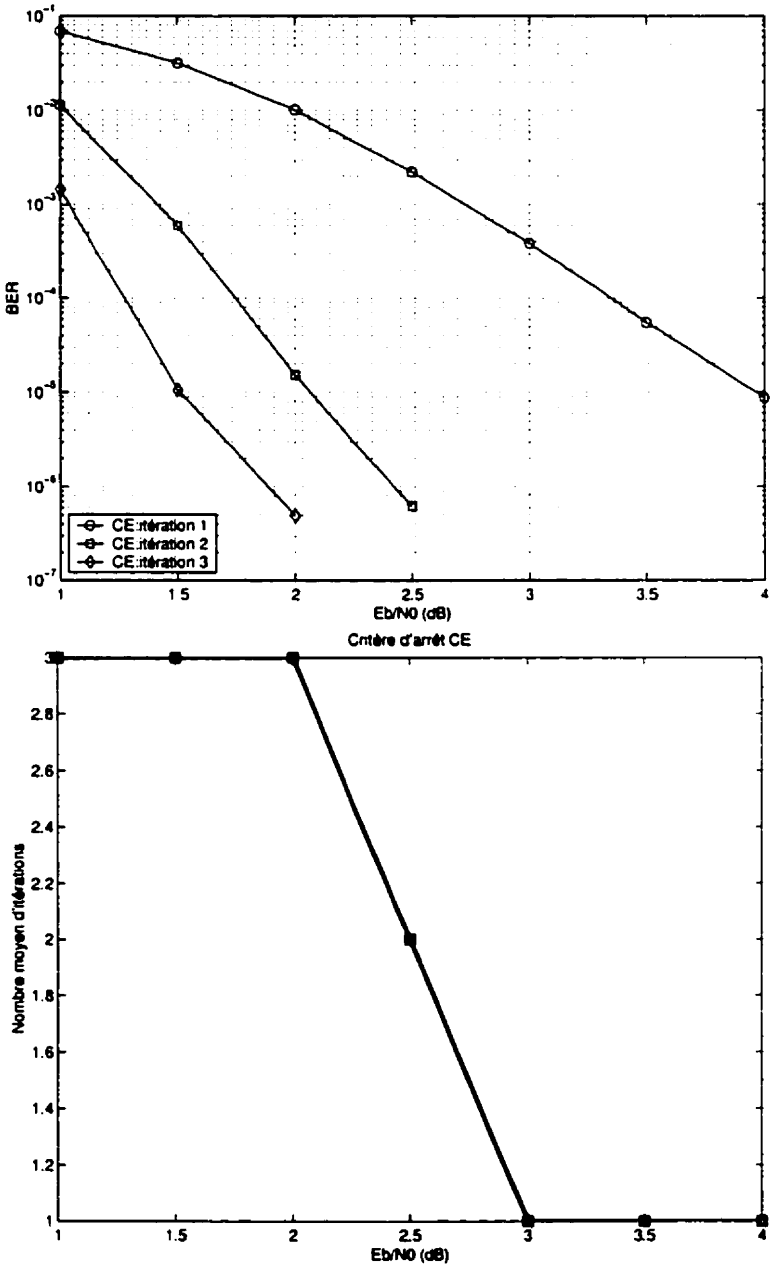


Figure 3.21: Performance du code turbo avec le critère d'arrêt CE dans un canal AWGN avec $T_n(i)/T_2(1) = 10^{-4}$, $K = 5$, $G = (1, 35/23)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 900

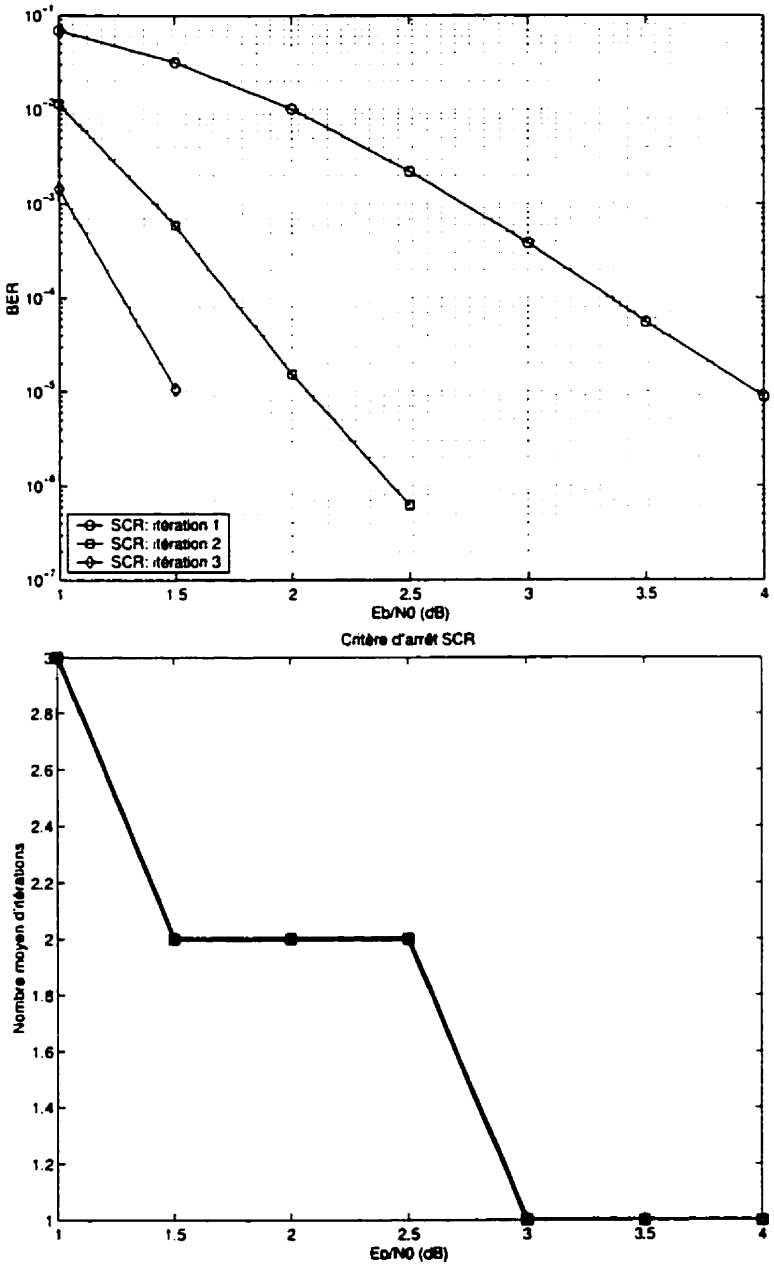


Figure 3.22: Performance du code turbo avec le critère d'arrêt SCR dans un canal AWGN avec $\frac{C^n(i)}{N} = 0.005$, $K = 5$, $G = (1, 35/23)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 900

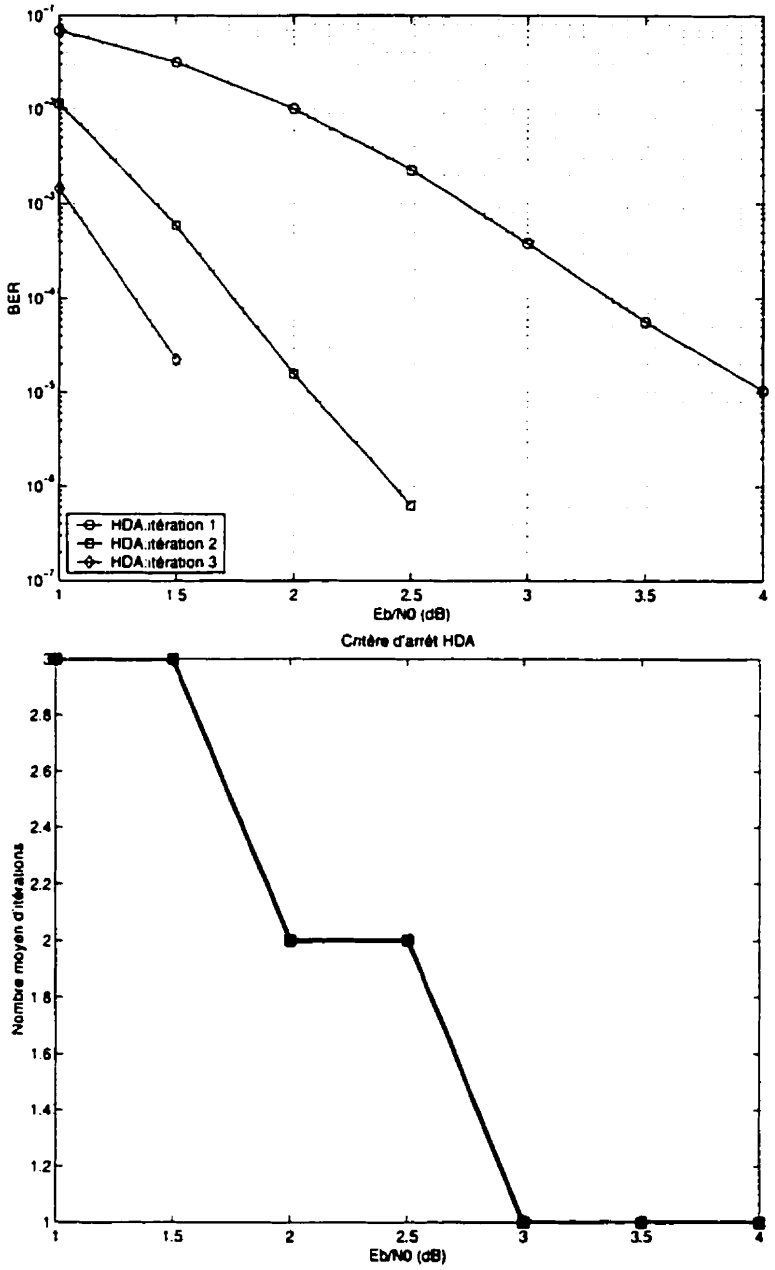


Figure 3.23: Performance du code turbo avec le critère d'arrêt HDA dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 900

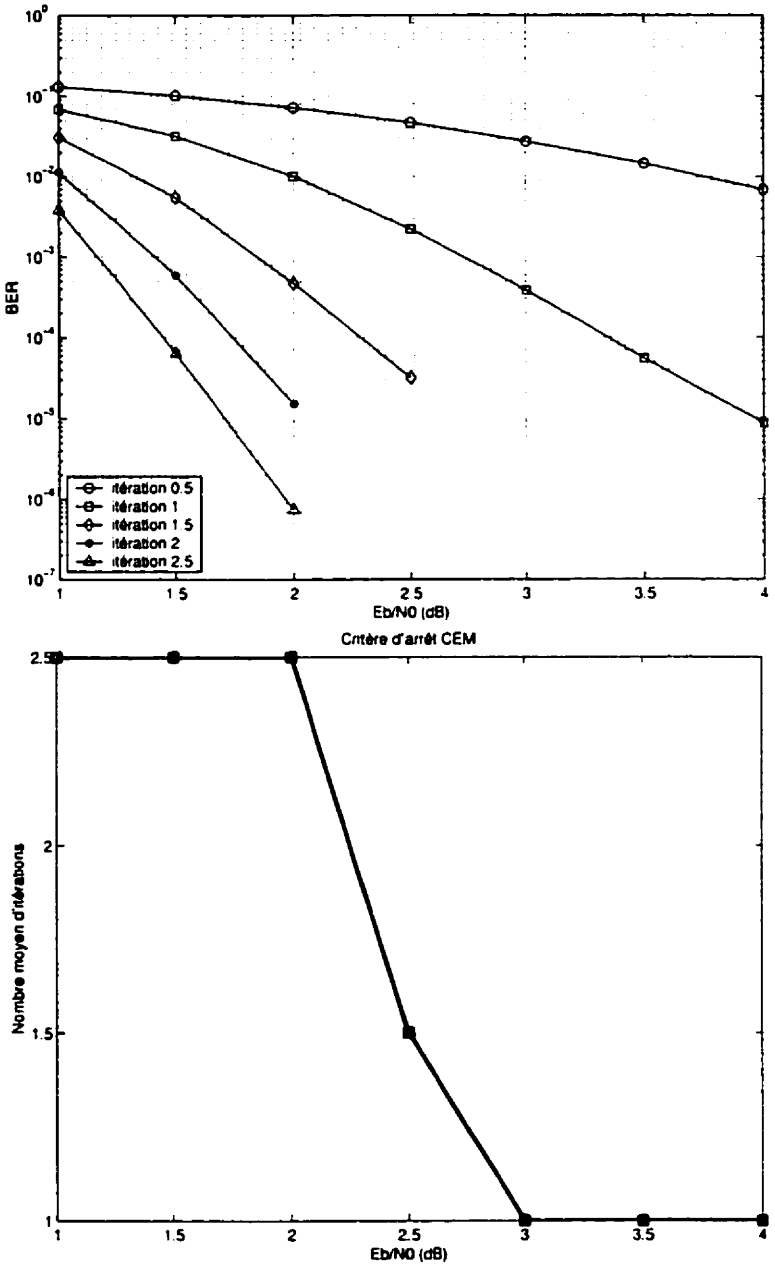


Figure 3.24: Performance du code turbo avec le critère d'arrêt CEM dans un canal AWGN avec $T_n(i)/T_2(1) = 10^{-4}$, $K = 5$, $G = (1, 35/23)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 900

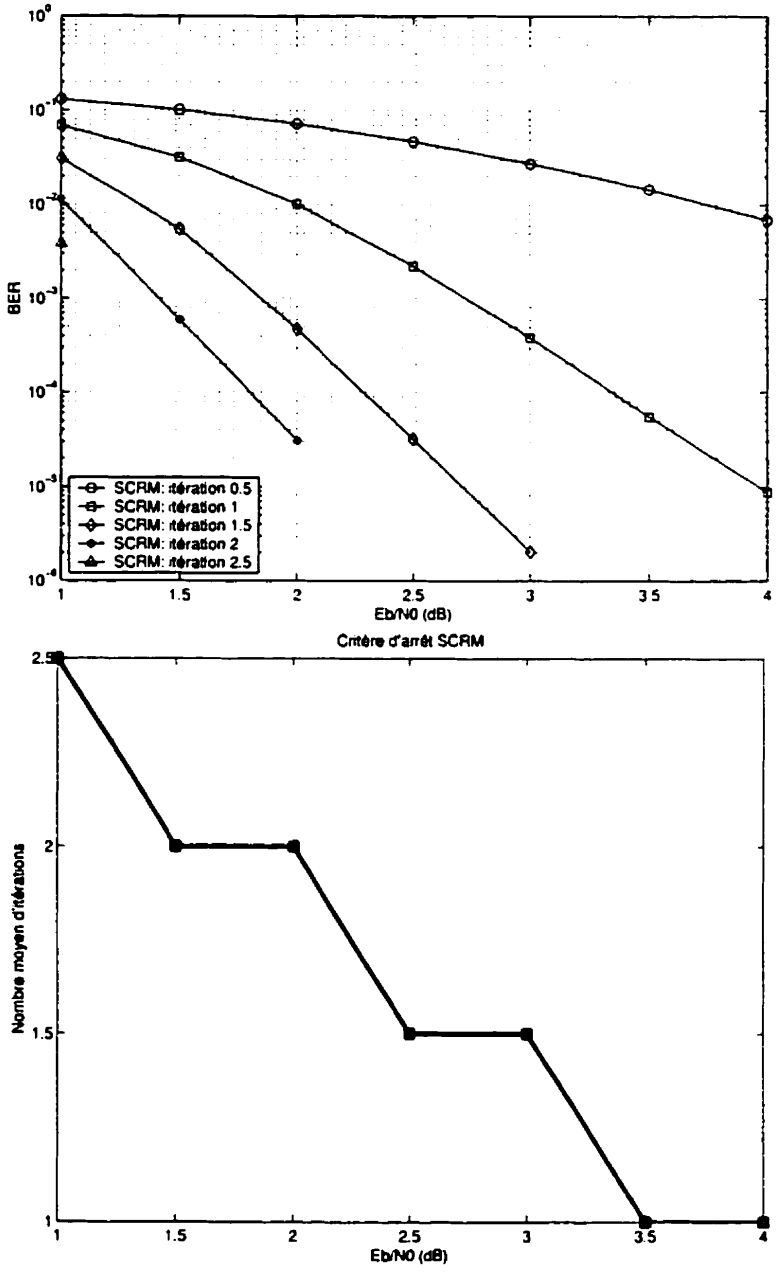


Figure 3.25: Performance du code turbo avec le critère d'arrêt SCRM dans un canal AWGN avec $\frac{C^n(i)}{N} = 0.005$, $K = 5$, $G = (1, 35/23)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 900

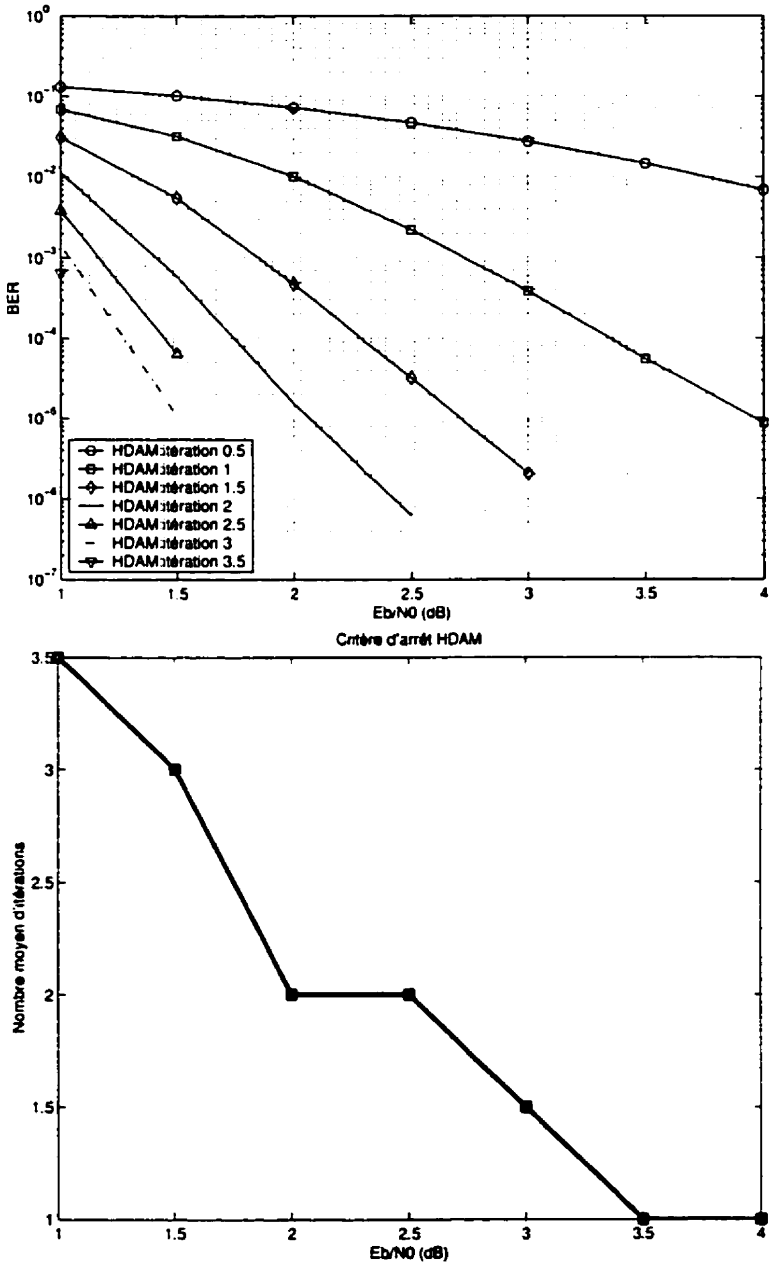


Figure 3.26: Performance du code turbo avec le critère d'arrêt HDAM dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 900

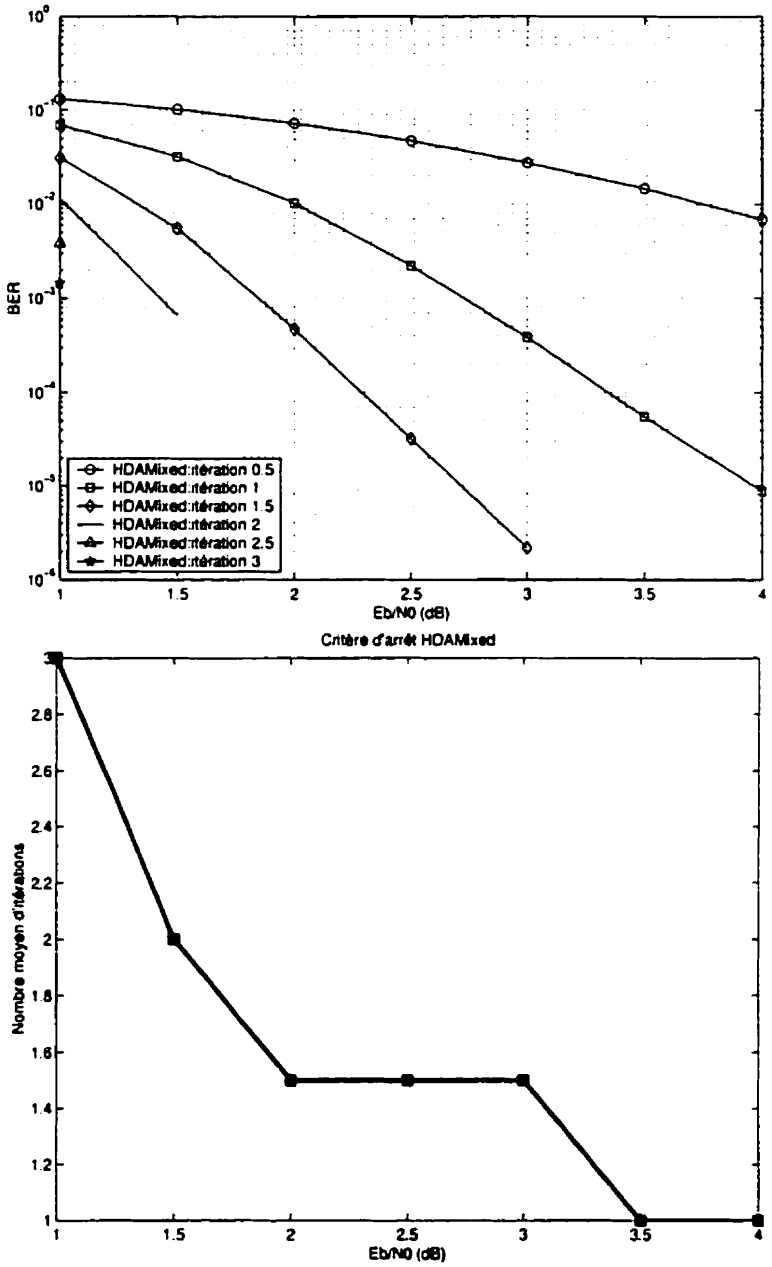


Figure 3.27: Performance du code turbo avec le critère d'arrêt HDAMixed dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 900

Chapitre 4

Décodage parallèle des codes turbo

4.1 Introduction

Dans le chapitre 3, nous avons introduit la technique de décodage turbo. En particulier, nous avons présenté l'algorithme de décodage utilisé. Dans ce chapitre, nous allons évoquer le problème de délai du décodage qui est parmi les facteurs les plus importants dans un système de communication numérique. Dans le cas du code turbo ce délai est très important par rapport à d'autres types de systèmes de décodage. Ainsi, nous allons présenter une solution pour diminuer ce délai qui dépend surtout de la version de l'algorithme MAP et de la taille de l'entrelaceur utilisé.

4.2 Inconvénient du décodage série des codes turbo

Dans la version originale décrite par Berrou *et al.* en 1993 [8], le décodage turbo s'effectue-en mode série comme nous venons de le voir dans le chapitre 3.

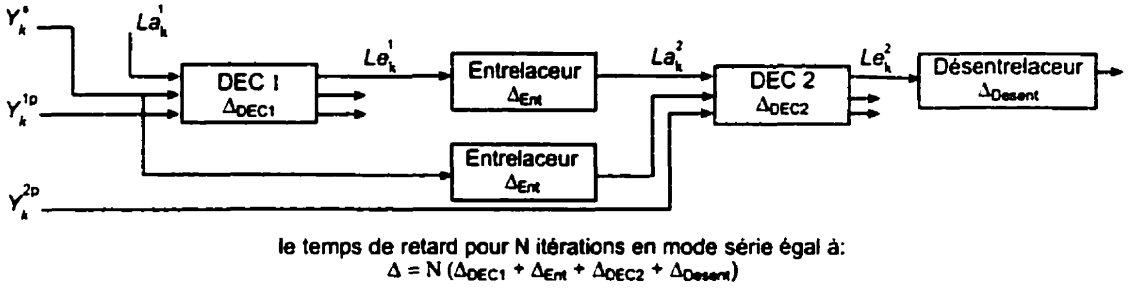


Figure 4.1: Ligne de retard de décodage turbo en mode série

En analysant la figure 4.1, nous pouvons remarquer que ce mode de décodage présente les inconvénients suivants :

1. Un grand nombre d'itérations qui entraînent de grands retards surtout pour des blocs très larges :

   Plus la taille de l'entrelaceur est grande plus les délais Δ_{Ent} et Δ_{Desent} sont importants. Pour N itérations, le délai de décodage global est égal à $\Delta = N.(\Delta_{DEC1} + \Delta_{Ent} + \Delta_{DEC2} + \Delta_{Desent})$. Puisque d'une part les tailles de l'entrelaceur et du délaceur sont les mêmes et que d'autre part les deux codeurs CRS sont identiques, alors $\Delta_{Ent} = \Delta_{Desent}$ et $\Delta_{DEC1} = \Delta_{DEC2} = \Delta_{DEC}$ de sorte que nous obtenons $\Delta = 2N.(\Delta_{DEC} + \Delta_{Ent})$.

2. Le fait qu'un décodeur fonctionne et que l'autre attende l'information provoque une utilisation non optimale des ressources :

   A part la première moitié de l'itération 1, chaque décodeur doit attendre une durée égale à $\Delta \simeq \Delta_{DEC} + \Delta_{Ent}$ avant de pouvoir commencer à décoder.

3. Performances limitées :

Chaque décodeur utilise une information qui n'est pas à la disposition de l'autre pour générer une information de fiabilité et à la fin une simple quantification de la sortie LLR de décodeur 2 est réalisée pour obtenir les bits décodés.

4.3 Motivation et solution

Examinons maintenant, le fait de mettre le DEC2 avant le DEC1 (c'est-à-dire nous commençons le décodage avec la parité 2). La figure 4.2 illustre cette architecture qui est un peu différente de l'ancienne présentée précédemment dans la figure 4.1. Nous désignons cette nouvelle architecture par ARCH2.

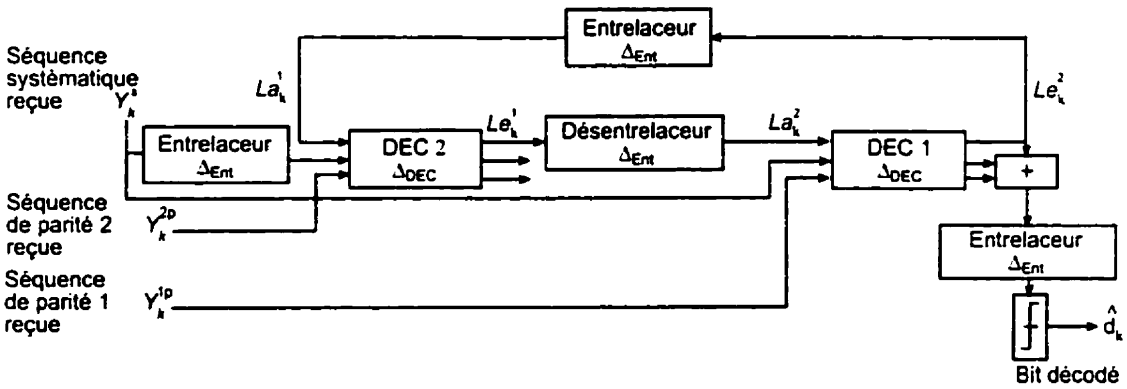


Figure 4.2: Schéma de décodage turbo avec DEC2 avant le DEC1

L'idée de mettre le DEC2 avant le DEC1 n'a pas été adoptée par les fondateurs du code turbo, bien que nous obtenons presque les mêmes performances, est due au fait qu'il y a des délais supplémentaires par rapport à l'ancienne architecture. En effet, le schéma exact de l'encodeur turbo utilisé dans l'ancienne architecture est illustré dans la figure 4.3.

Nous remarquons dans cette figure, qu'il suffit de retarder les bits d'information d'un certain temps égal à $T = \Delta_{DEC}$, délai de décodage du premier décodeur

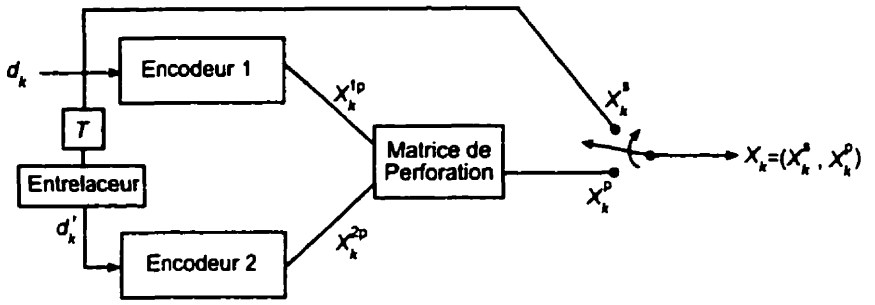


Figure 4.3: Schéma de l'encodeur turbo réel

(DEC1), avant de les faire passer à travers l'entrelaceur pour assurer la synchronisation lors du décodage. Cependant dans l'ARCH2, il faut retirer ce temps de retard et retarder les symboles d'entrée ou de sortie du l'encodeur 1 d'un délai égal au temps de traitement de tous les entrelaceurs plus le temps de décodage du DEC2. c'est à dire. $T = \Delta_{DEC} + 3.\Delta_{Ent}$.

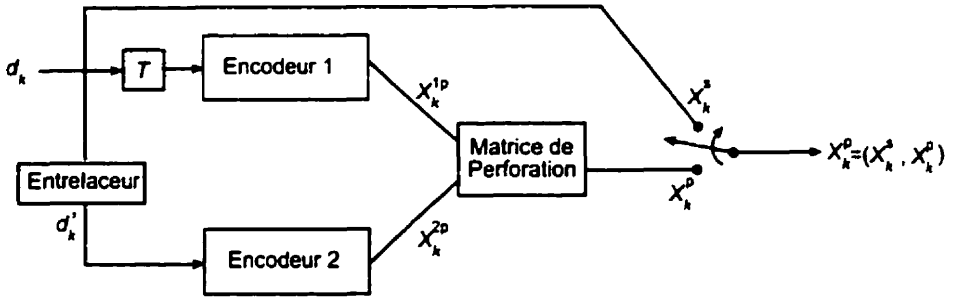


Figure 4.4: Schéma de l'encodeur turbo correspondant au schéma de décodage de la figure 4.2

Si nous ignorons ce facteur de délai avec l'ARCH2, nous retrouverons les mêmes inconvénients mentionnés ci-dessus, mais nous remarquerons que la performance peut aussi bien s'améliorer ou se dégrader par rapport à l'architecture traditionnelle de décodage comme l'indique la figure 4.5.

Alors, pour mieux profiter des deux décodeurs et retirer les inconvénients de

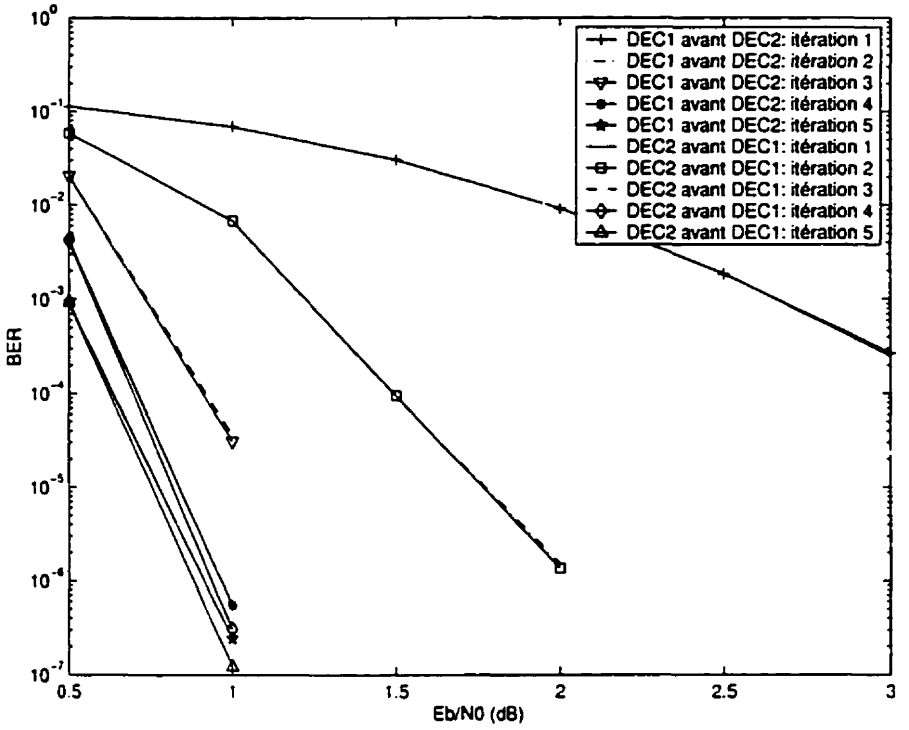


Figure 4.5: Effet de l'ordre des DECs dans le mode série sur la performance des codes turbo avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 1/3$, canal AWGN et taille d'entrelaceur aléatoire 4096

l'utilisation du mode série lors du décodage, il serait intéressant d'utiliser un modèle où les deux décodeurs fonctionnent en parallèle et en même temps.

4.4 Décodage Parallèle des codes turbo

4.4.1 Principe

Le décodage en mode parallèle a été proposé par Divsalar et Pollara [14]. Au niveau codage, nous avons deux codeurs CRS concaténés en parallèle et séparés par un entrelaceur comme dans la figure 4.4. Les séquences reçues sont : $\mathbf{Y}^s$, $\mathbf{Y}^{1p}$ et $\mathbf{Y}^{2p}$ qui correspondent respectivement aux séquences émises systématique, parité 1 et parité 2. Les deux processus de décodage s'effectuent simultanément.

Le décodage se fait après un certain nombre d'itérations. Dans notre cas, contrairement au décodage série où l'itération prend fin à la sortie du deuxième décodeur, l'itération s'achève à la sortie de l'un des deux DECs simultanément puisqu'ils sont montés en parallèle comme l'indique la figure 4.6.

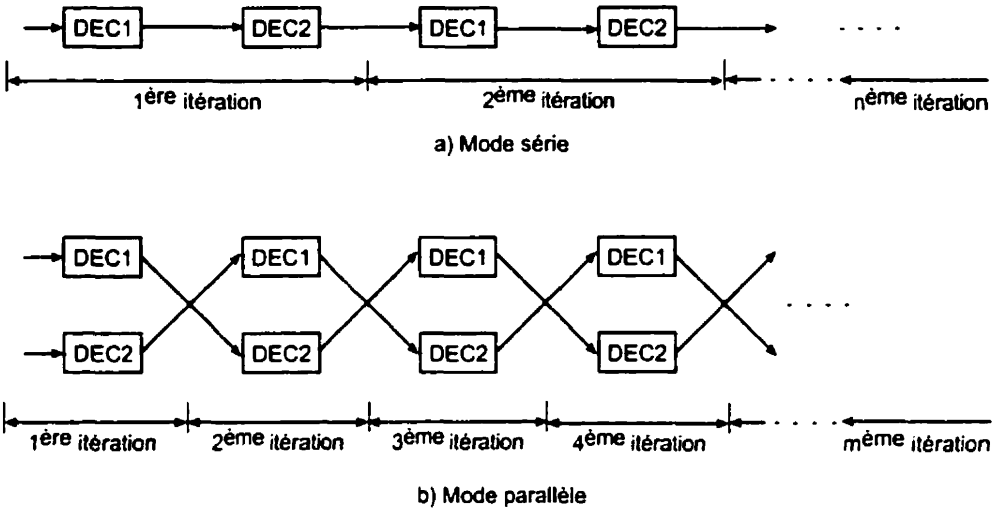


Figure 4.6: Structure de décodage turbo en modes série et parallèle

Considérons maintenant, le décodage turbo en mode parallèle. A l'instant k et pour chaque itération i ($1 \leq i \leq M$, avec M nombre maximal d'itérations), les entrées du DEC1 sont les symboles Y_k^s, Y_k^{1p} et l'information à priori $La_k^1(i)$ et ceux du DEC2 sont les symboles Y_k^{1s} (symboles de l'information systématique entrelacés), Y_k^{2p} et $La_k^2(i)$. Les valeurs de $La_k^1(1)$ pour le DEC1 et $La_k^2(1)$ pour le DEC2 sont égales à zéro durant la première itération puisque les bits d'information sont équiprobables pour les deux décodeurs. A la sortie du DEC1 (respectivement DEC2), nous trouvons la valeur du rapport de vraisemblance $\Lambda_k^1(i)$ (respectivement $\Lambda_k^2(i)$) et l'information extrinsèque $Le_k^1(i)$ (respectivement $Le_k^2(i)$).

Si le critère d'arrêt en mode parallèle n'est pas encore atteint, La séquence

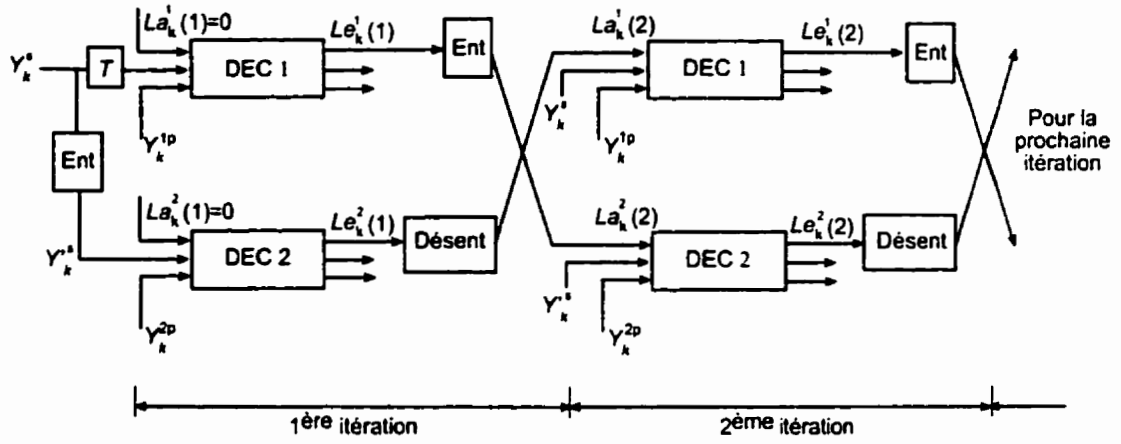


Figure 4.7: Décodage turbo en mode parallèle

d'information extrinsèque $\mathbf{Le}^1(i)$ (respectivement $\mathbf{Le}^2(i)$) sera entrelacée (respectivement délacée) avant d'être fournie au DEC2 (respectivement DEC1) comme une information à priori $\mathbf{La}^2(i+1)$ (respectivement $\mathbf{La}^1(i+1)$) durant la prochaine itération.

Sinon, il est possible d'arrêter le processus de décodage et déterminer la valeur de vraisemblance finale. En fait, deux valeurs de fiabilité sont disponibles à chaque itération i qui sont:

$$\Lambda_k^1(i) = \frac{2}{\sigma^2} Y_k^s + La_k^1(i) + Le_k^1(i) \tag{4.1}$$

$$\Lambda_k^2(i) = \frac{2}{\sigma^2} Y_k'^s + La_k^2(i) + Le_k^2(i) \tag{4.2}$$

Ainsi, pour trouver la valeur de fiabilité finale $\Lambda_k(i)$, il faut d'abord délacer $\mathbf{\Lambda}^2(i)$ (la séquence obtenue est $\dot{\mathbf{\Lambda}}^2(i)$) puis procéder comme suit:

$$\Lambda_k(i) = \begin{cases} \Lambda_k^1(i), & \text{si } |\Lambda_k^1(i)| \geq |\dot{\Lambda}_k^2(i)| \\ \\ \dot{\Lambda}_k^2(i), & \text{autrement} \end{cases} \tag{4.3}$$

La règle de décision qui détermine le symbole le plus vraisemblable est la même qu'au chapitre 3, c'est-à-dire:

$$\tilde{d}_k = \begin{cases} 1, & \text{si } \Lambda_k(i) \geq 0 \\ \\ 0, & \text{autrement} \end{cases} \tag{4.4}$$

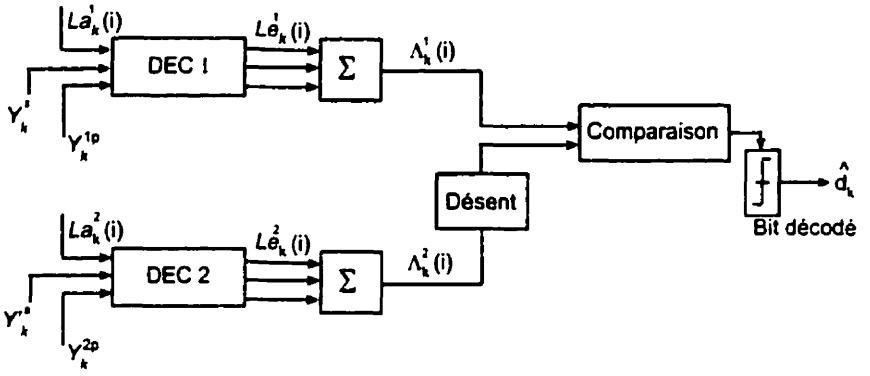


Figure 4.8: Dernière itération de décodage turbo en mode parallèle

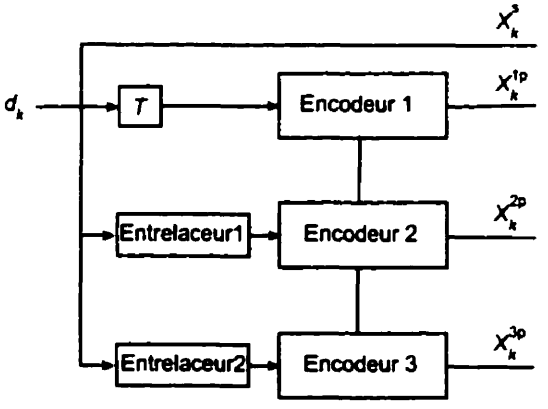


Figure 4.9: Encodeur Turbo avec 3 codeurs CRS

Il faut mentionner que le principe de décodage parallèle, reste le même pour un code turbo de différentes valeurs de taux de codage $R_{gp} \geq 1/3$. Cependant, pour

les valeurs plus faibles: exemple $R_{gp} = 1/4$, la partie codage a au moins 3 codeurs CRS en parallèle comme l'indique la figure 4.9, un décodeur en particulier peut avoir plus d'une information supplémentaire (information à priori). En d'autres termes, sa capacité de correction peut être améliorée et ainsi la performance globale de décodage augmentera après chaque itération.

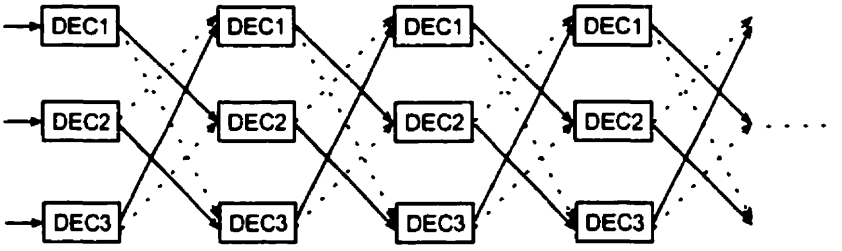


Figure 4.10: Décodeur turbo parallèle avec une seule information à priori pour chaque DEC

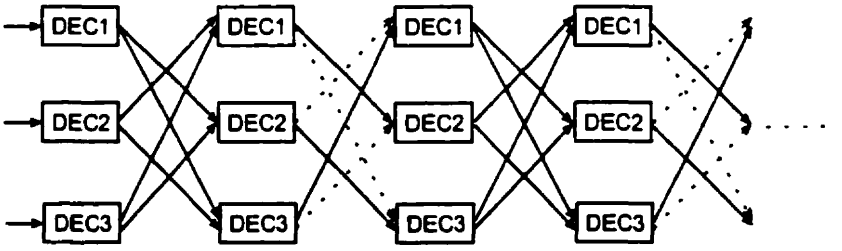


Figure 4.11: Décodeur turbo parallèle avec une seule ou deux informations à priori pour chaque DEC

Dans ce cas, il existe différentes façons pour passer les informations extrinsèques d'un décodeur à un autre:

- ***Mode Unique:*** Chaque décodeur reçoit une seule information à priori. Comme l'indique la figure 4.10 et à chaque itération, on active les traits continus ou bien les traits pointillés.
- ***Mode Alterné:*** Durant les itérations impaires chaque décodeur reçoit au moins deux informations à priori. Par contre pour les itérations paires une

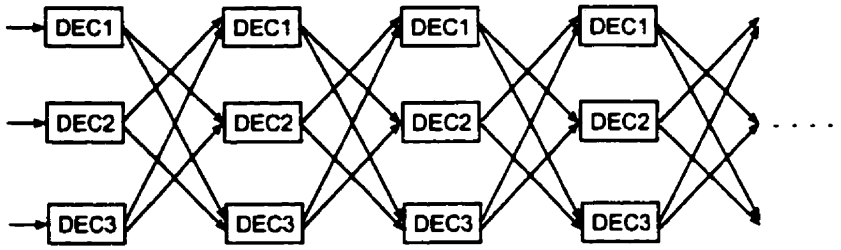


Figure 4.12: Décodeur turbo parallèle avec deux informations à priori pour chaque DEC

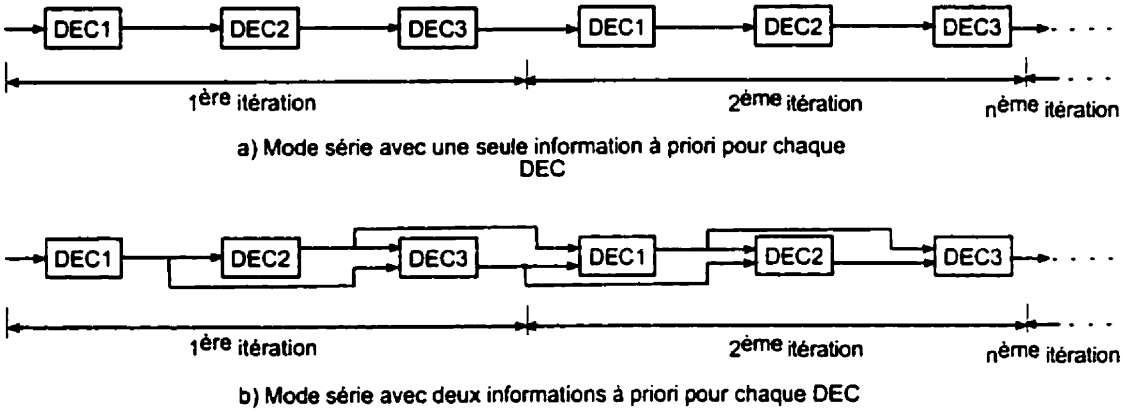


Figure 4.13: Schémas du décodeur turbo série avec un taux de codage $R_{gp} = 1/4$

seule information à priori est reçue par décodeur. La figure 4.11 illustre bien ce mode.

- ***Mode Total:*** Dans toutes les itérations chaque décodeur dispose d'au moins deux informations à priori comme l'indique la figure 4.12.

Dans le cas du code turbo en mode série avec un taux de codage $R_{gp} = 1/4$, il existe aussi plusieurs façons de faire passer les informations extrinsèques d'un décodeur à un autre. La figure 4.13 montre que dans (a) une seule information à priori en entrée pour chaque décodeur alors que dans (b) chaque DEC reçoit deux différentes informations à priori. Dans ces deux cas, il est clair que chaque

décodeur doit atteindre un temps double de celui en mode série avec un taux de codage $R_{gp} \geq 1/3$.

4.4.2 Synchronisation et ligne de retard du décodage parallèle

1. ***Synchronisation :***

   En suivant un raisonnement similaire à celui des sections précédentes et vu que dans l'architecture parallèle les deux décodeurs doivent fonctionner en même temps, la synchronisation entre les entrées du DEC1 et du DEC2 doit être assurée. Pour ce faire, nous devrons retarder les entrées du codeur 1 et celle du DEC1 d'un délai qui est égal au temps de traitement de l'entrelaceur. Donc, les figures 4.4 et 4.7 avec $T = \Delta_{Ent}$ correspondent bien respectivement à l'encodeur et décodeur parallèle turbo. Ce même principe de synchronisation est utilisé dans l'architecture parallèle avec des taux de codage $R_{gp} < 1/3$.

2. ***ligne de retard :***

   La valeur de Δ_{DEC}: temps de processus de décodage (respectivement Δ_{Ent}: temps de traitement de l'entrelaceur), dépend de l'algorithme APP et de la longueur de contrainte des codeurs CRS (respectivement de la taille et de type de l'entrelaceur). Par conséquent, il est quelque peu difficile de donner un ordre de grandeur entre Δ_{DEC} et Δ_{Ent}.

   Toutefois, nous allons utiliser les figures 4.1 et 4.14 pour pouvoir comparer le délai des deux modes de décodage présentés. Nous pouvons donc déduire qu'un délai $\Delta_{parallèle}$ de N itérations en mode parallèle pour les taux de codage $R_{gp} \geq 1/3$ est égal à:

$$\Delta_{parallèle} = \frac{\Delta_{série}}{2} + \Delta_{Ent} \tag{4.5}$$

où $\Delta_{série}$ est le délai de N itérations en mode série avec le même taux de codage que celui du mode parallèle.

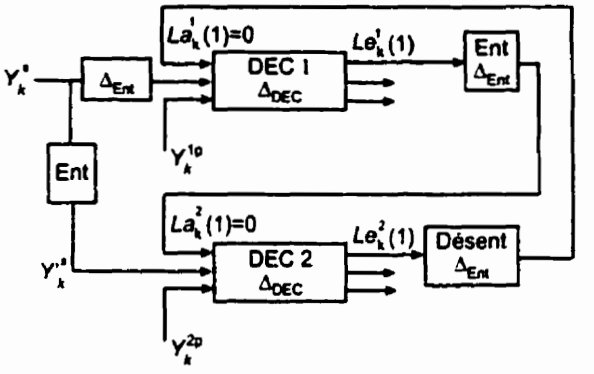


Figure 4.14: Ligne de retard de décodage turbo en mode parallèle

Dans le cas de $R_{gp} = 1/4$, ce délai s'exprime par:

$$\Delta_{parallèle} = \frac{\Delta_{série}}{3} + \Delta_{Ent} \tag{4.6}$$

4.4.3 Résultats de simulation

Avant d'entamer l'analyse des performances du décodage parallèle, il est nécessaire d'indiquer que dans l'ensemble des simulations réalisées, nous avons utilisé l'algorithme Log-MAP dans un canal AWGN pour des valeurs de taux de codage $R_{gp} = 1/2$, $1/3$ et $1/4$.

Il faut signaler aussi que nous allons utiliser les approximations suivantes:

- pour les taux de codage $R_{gp} = 1/2$ et $1/3$: le délai de 2 itérations en mode parallèle est égal au délai d'une itération en mode série.
- Alors que pour $R_{gp} = 1/4$, le délai de trois itérations en mode parallèle est égal au délai d'une itération en mode série.

4.4.3.1 Modes parallèle et série pour $R_{gp} = 1/2$

Pour le taux de codage $R_{gp} = 1/2$, la figure 4.15 montre que le mode parallèle fournit une légère amélioration de la performance par rapport au mode série mais avec le même temps de décodage.

Donc, il serait inutile d'utiliser le mode parallèle pour des taux de codage supérieurs à $1/2$ puisque d'une part le gain de codage est très faible et d'autre part l'implémentation matérielle demande plus de ressources que le mode série.

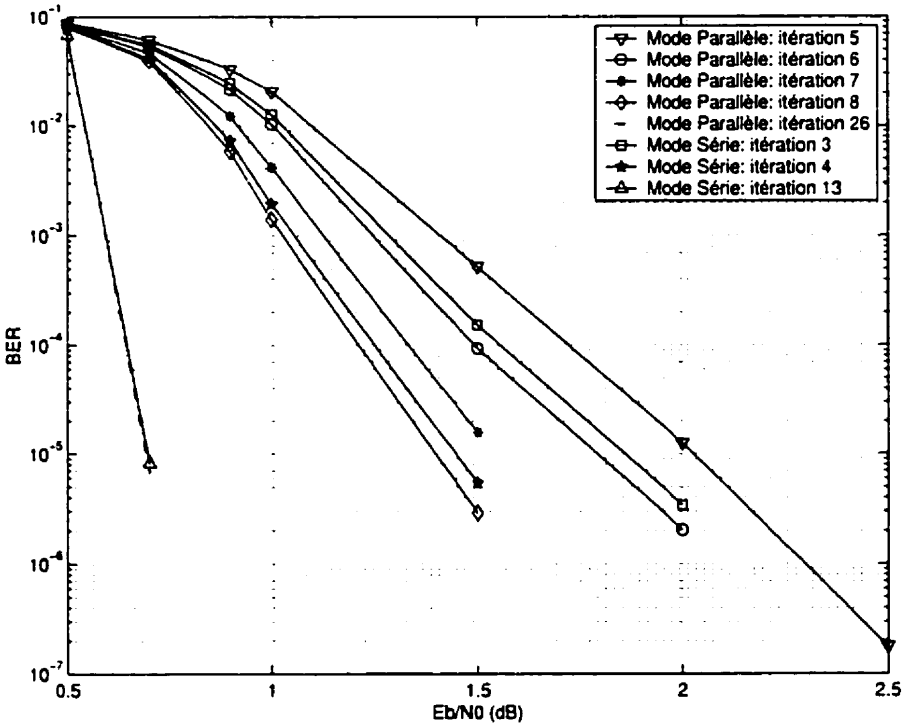


Figure 4.15: Performance des codes turbo en modes série et parallèle dans un canal AWGN avec $K = 5$, $G = (1, 21/37)$, $R_{gp} = 1/2$ et taille d'entrelaceur aléatoire 65536

4.4.3.2 Modes parallèle et série pour $R_{gp} = 1/3$

Avec le taux de codage $R_{gp} = 1/3$, le mode parallèle fournit une meilleure performance que le mode série avec un délai de décodage plus faible. En effet, la figure 4.16 montre que pour un nombre élevé d'itérations, le mode parallèle nous permet de gagner un ou demi du temps d'une itération en mode série (l'itération 24 du mode parallèle atteint presque les mêmes performances que l'itération 13 en mode série). Par ailleurs, au delà du rapport $E_b/N_0 = 0.7\ dB$, l'architecture parallèle permet aussi d'économiser la moitié du temps d'une itération en mode série: un $BER < 10^{-5}$ est obtenu après 9 itérations en mode parallèle alors qu'il nous fallait 5 itérations en mode série pour atteindre le même but.

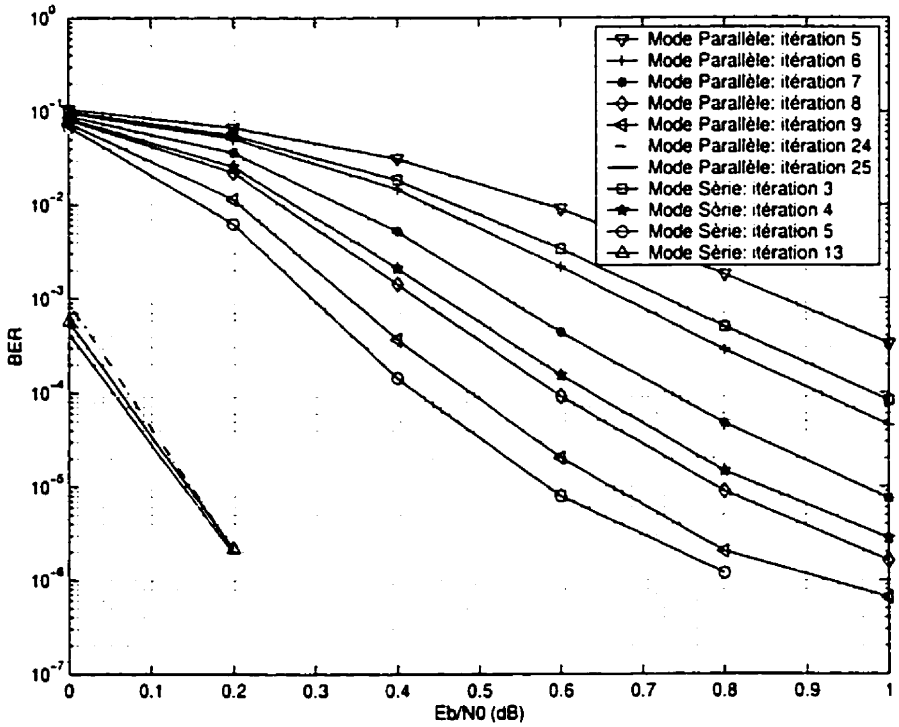


Figure 4.16: Performance des codes turbo en modes série et parallèle dans un canal AWGN avec $K = 5$, $G = (1, 21/37)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 65536

4.4.3.3 Modes parallèle et série pour $R_{gp} = 1/4$

Les figures 4.17, 4.18, 4.19 et 4.20, montrent aussi bien dans le cas du mode parallèle que dans le mode série, que plus le nombre des séquences d'information à priori est grand à l'entrée d'un tel décodeur, plus les performances sont meilleures. Ainsi, nous pouvons classer les différents modes parallèles du plus mauvais au meilleur comme suit: mode unique, mode alterné et mode total. Dans la suite, nous utilisons le mode total pour tous les résultats de simulation.

La figure 4.21 illustre les performances du code turbo avec les modes parallèle et série avec un taux de codage $R_{gp} = 1/4$. Dans ce cas, le mode parallèle fournit une meilleure performance que le mode série avec un délai de décodage très faible. En effet, les performances à partir de la $4^{ème}$ itération en mode parallèle sont meilleures que celles avec le mode série après 5 itérations. Il ne faut pas oublier que le temps de 4 itérations en mode parallèle égal à celui de 4/3 itérations en mode série. Donc, nous aurons un gain de temps presque égal à 15/4.

Nous remarquons aussi que:

- Avec le mode parallèle, plus le nombre d'itérations augmente, plus la performance est meilleure.
- Le mode parallèle sature moins vite que celui du mode série.
- Le mode parallèle s'approche plus de la limite de Shannon que le mode série.

Avec le taux de codage $R_{gp} = 1/4$, nous avons essayé d'utiliser l'algorithme Max-Log-MAP avec le mode parallèle et de comparer ces performances par la suite avec le mode série. La figure 4.22 illustre les résultats de simulation avec $K = 5$, $G = (1, 35/23)$ et une taille d'entrelaceur 1024. Nous remarquons que le mode parallèle avec l'algorithme Max-Log-MAP reste encore meilleur que le mode série utilisant l'algorithme Log-MAP. En effet, juste à partir de la deuxième itération du mode parallèle, l'algorithme Max-Log-MAP fournit une meilleure performance et

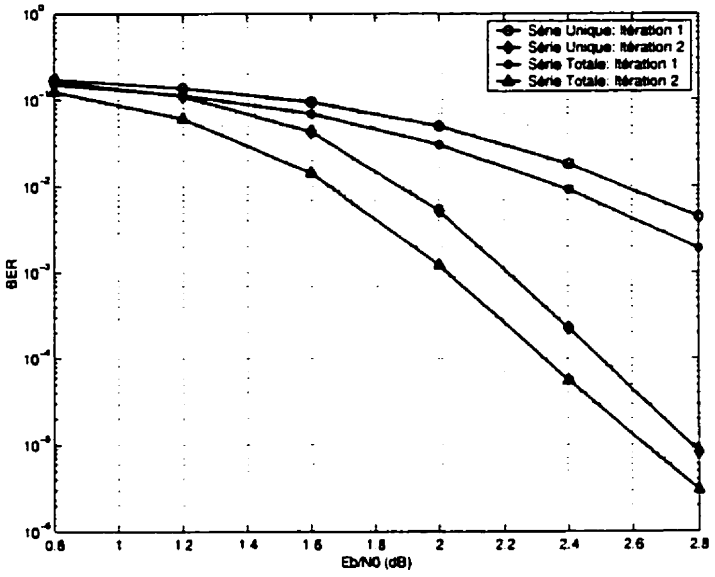


Figure 4.17: Performance des codes turbo en modes parallèle unique et alterné dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, $R_{gp} = 1/4$ et taille d'entrelaceur aléatoire 1024

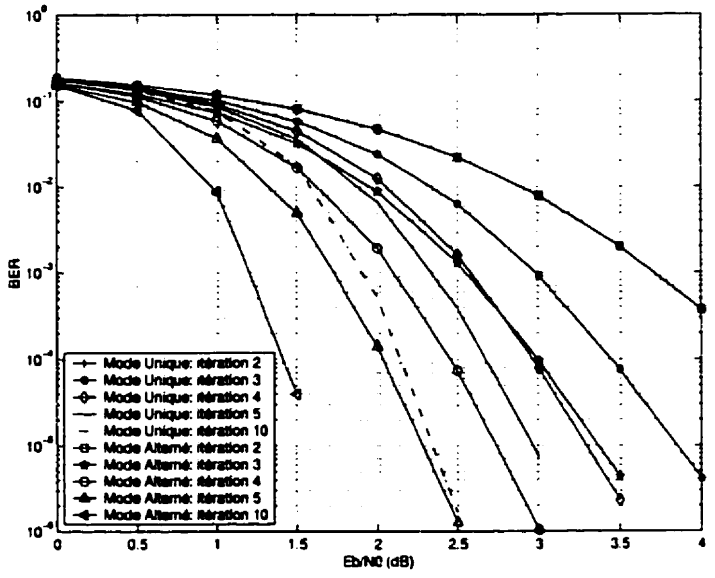


Figure 4.18: Performance des codes turbo en modes parallèle unique et alterné dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, $R_{gp} = 1/4$ et taille d'entrelaceur aléatoire 1024

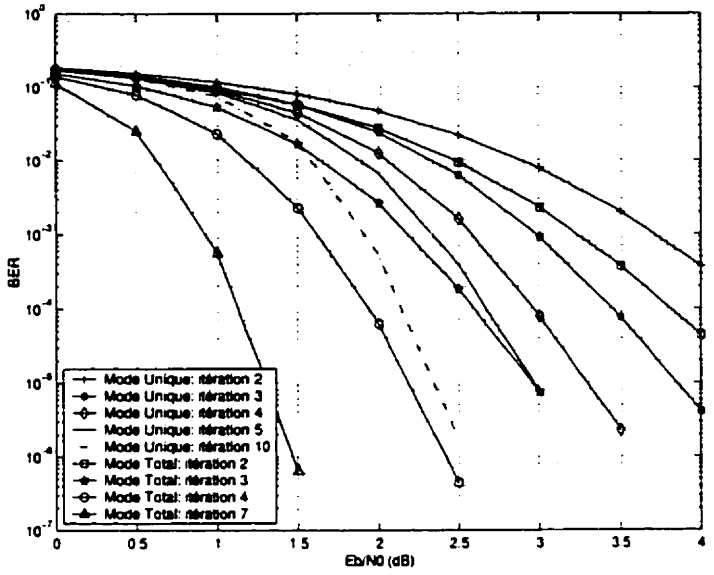


Figure 4.19: Performance des codes turbo en modes parallèle unique et Total dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, $R_{gp} = 1/4$ et taille d'entrelaceur aléatoire 1024

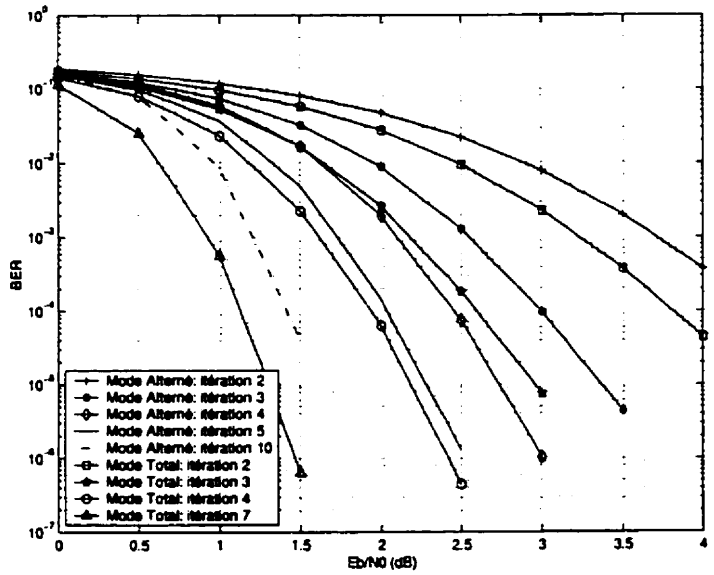


Figure 4.20: Performance des codes turbo en modes parallèle alterné et Total dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, $R_{gp} = 1/4$ et taille d'entrelaceur aléatoire 1024

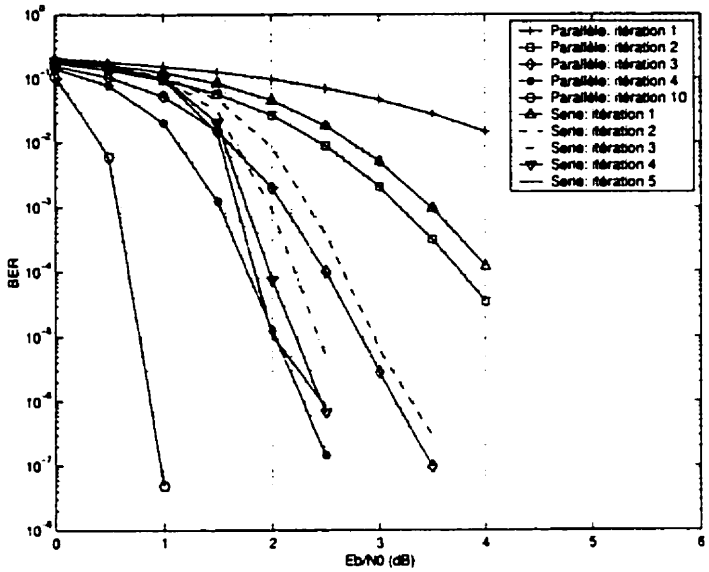


Figure 4.21: Performance des codes turbo en mode série et parallèle dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, $R_{gp} = 1/4$ et taille d'entrelaceur aléatoire 3600

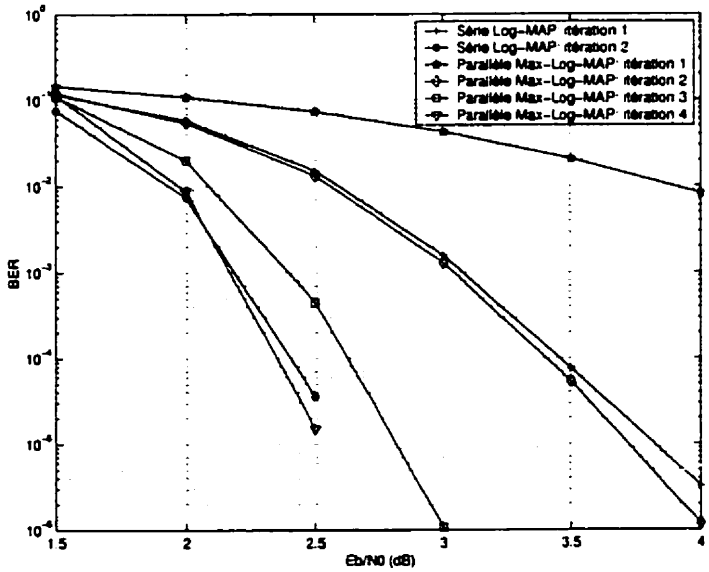


Figure 4.22: Performance du Max-Log-MAP en mode parallèle et du Log-MAP en mode série dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 1/4$ et taille d'entrelaceur aléatoire 1024

moins de délai que ceux obtenus avec l'algorithme Log-MAP à la première itération en mode série.

4.4.4 Critères d'arrêt

Nous avons présenté dans le chapitre 3 les différents critères d'arrêt du processus de décodage turbo en mode série. Nous pouvons les appliquer aussi dans le mode parallèle mais avec une légère modification.

4.4.4.1 Entropie croisée (CE-P) pour le décodage parallèle

Nous redéfinissons la différence entre les informations extrinsèques à la $i^{\text{ème}}$ itération par:

$$\triangle Le_k(i) = |Le_k^2(i) - Le_k^1(i)|, i \geq 1 \tag{4.7}$$

Ainsi, l'entropie croisée $T(i)$ correspondant aux DEC1 et DEC2 à la $i^{\text{ème}}$ itération, peut être évaluée approximativement par:

$$T(i) \approx \sum_{k=1}^{N} \frac{(\triangle Le_k(i))^2}{e^{|\Lambda_k(i)|}}, i \geq 1 \tag{4.8}$$

Et

si $T(i)/T(1) = 10^{-2} \sim 10^{-4}$, nous devrons arrêter le décodage

Il est clair, que dans ce cas nous n'avons pas besoin d'espace mémoire pour garder une copie de $Le_k^1(i-1)$ ou bien $Le_k^2(i-1)$ mais le nombre d'opérations reste le même que celui dans le cas du mode série.

4.4.4.2 Critère d'arrêt SCR-P dans le mode parallèle

Soit $C(i)$ le nombre de changement de signes entre les éléments de la séquence d'information extrinsèque $\mathbf{Le}^1(i)$ et ceux de la séquence $\mathbf{Le}^2(i)$ à la $i^{\text{ème}}$ itération avec $i \geq 1$.

Le critère SCR dans le cas de décodage parallèle est encore plus simple. Il s'agit d'évaluer le rapport $\frac{C(i)}{N}$. S'il est inférieur à $\leq (0.005 \sim 0.03)$ alors le décodage peut être arrêté sans aucune dégradation de la performance.

Dans ce cas ci, l'unité de mémoire n'est pas utile.

4.4.4.3 Critère d'arrêt HDA-P dans le décodage parallèle

Contrairement au critère d'arrêt HDA vu au chapitre 3, la mémoire tampon qui garde des copies des valeurs de fiabilités précédentes sera inutile puisque nous comparons à chaque itération i, les signes des éléments de la séquence $\Lambda^1(i)$ et ceux de $\Lambda^2(i)$. Si nous trouvons presque les mêmes signes alors le décodage parallèle peut être arrêté.

4.4.4.4 Comparaisons de la performance des différents critères d'arrêt en mode parallèle

Le tableau 4.1. montre le nombre d'opérations nécessaires pour chaque critère d'arrêt en mode parallèle. Au contraire du mode série comme l'indique le tableau 3.1 du chapitre 3, les critères d'arrêt en mode parallèle n'ont pas besoin des mémoires.

Les figures 4.24, 4.26 et 4.28 montrent les performances du code turbo en mode parallèle utilisant respectivement les critères d'arrêt SCR-P avec $\frac{C(i)}{N} = 0.005$, HDA-P et CE-P avec $T(i)/T(1) = 10^{-4}$. Toutefois, avec les figures 4.25, 4.27 et 4.29, nous remarquons que le critère CE-P est le meilleur. Cependant, il demande plus d'opérations de calcul.

Tableau 4.1: Différents critères d'arrêt en mode parallèle

	CE-P	SCR-P	HDA-P
opérations	5N-1	3N-1	2N
mémoires	0	0	0

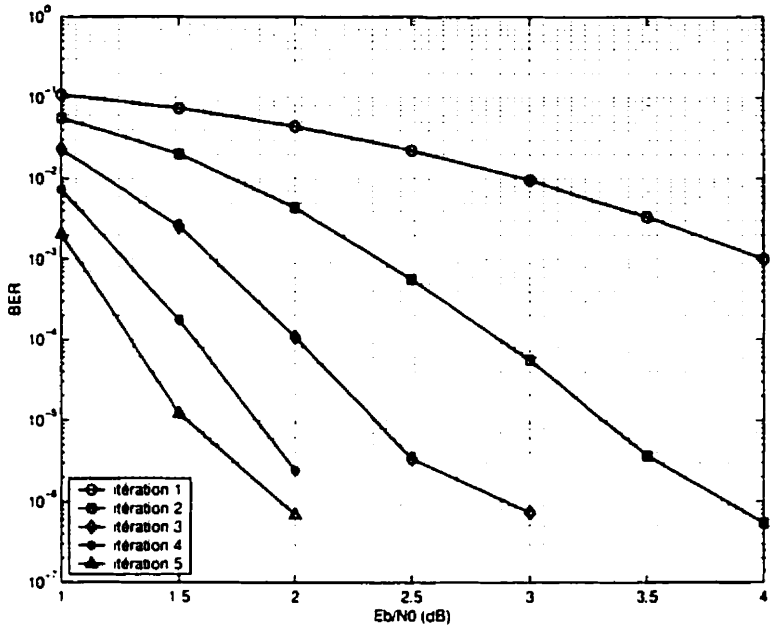


Figure 4.23: Performance du code turbo en mode parallèle sans critère d'arrêt dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 1024

4.5 Conclusion

Avec la nouvelle architecture de décodage turbo, nous avons vu qu'il est possible d'améliorer la performance tout en diminuant le délai de traitement. Cependant, pour des taux de codage élevés $\geq 1/2$, il est inutile d'utiliser l'architecture parallèle, puisque nous retrouvons presque les mêmes performances obtenues avec le décodage série.

Dans le prochain chapitre, nous allons s'intéresser au mode série du code turbo afin de présenter une nouvelle technique de perforation qui permet d'augmenter le taux de codage et d'avoir une meilleure performance.

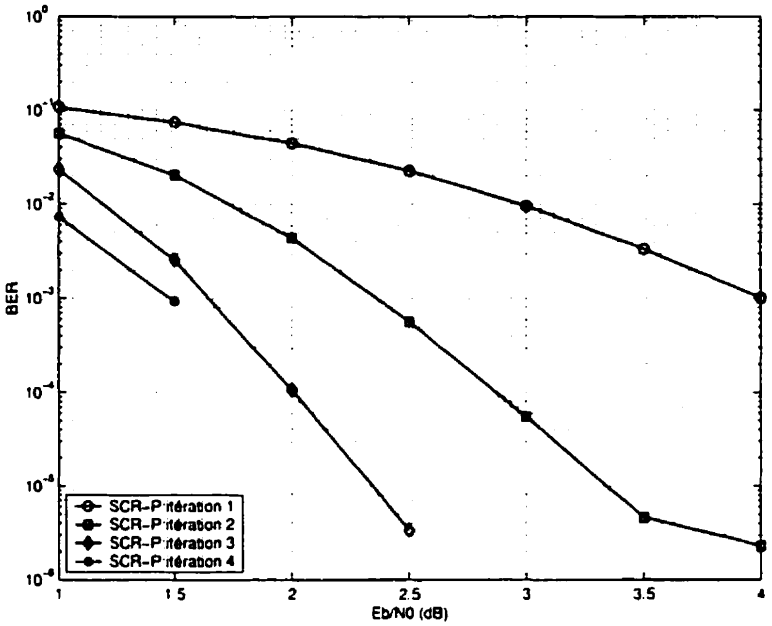


Figure 4.24: Critère d'arrêt SCR-P: performance du code turbo en mode parallèle dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 1024

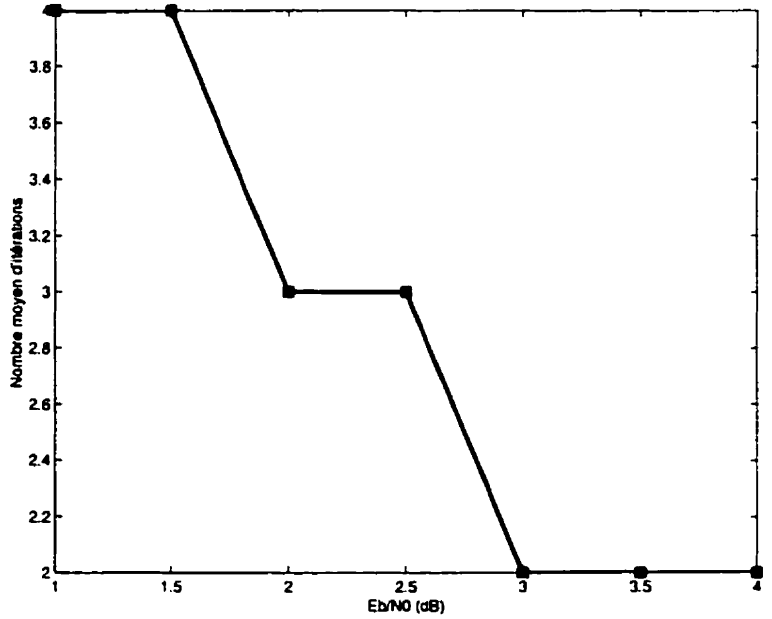


Figure 4.25: Nombre Moyen d'itérations du code turbo en mode parallèle utilisant le critère de la figure 4.24

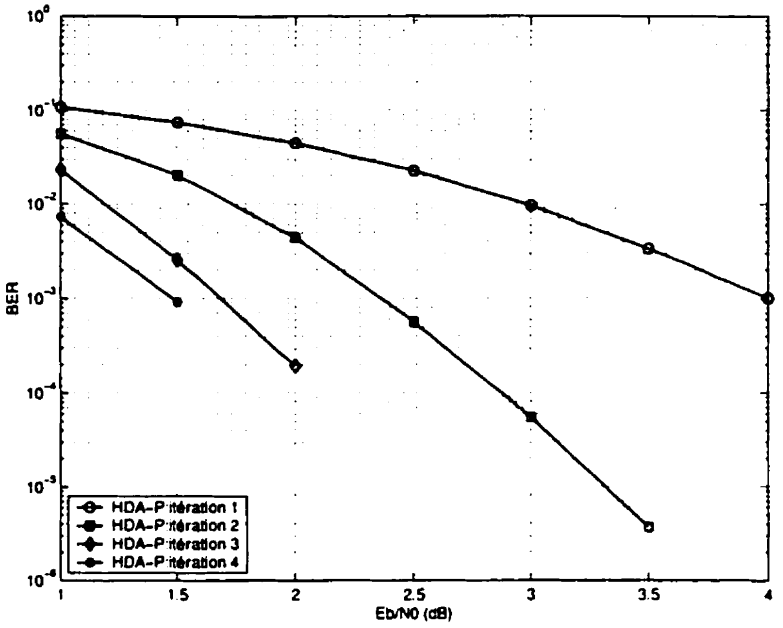


Figure 4.26: Critère d'arrêt HDA-P: performance du code turbo en mode parallèle dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 1024

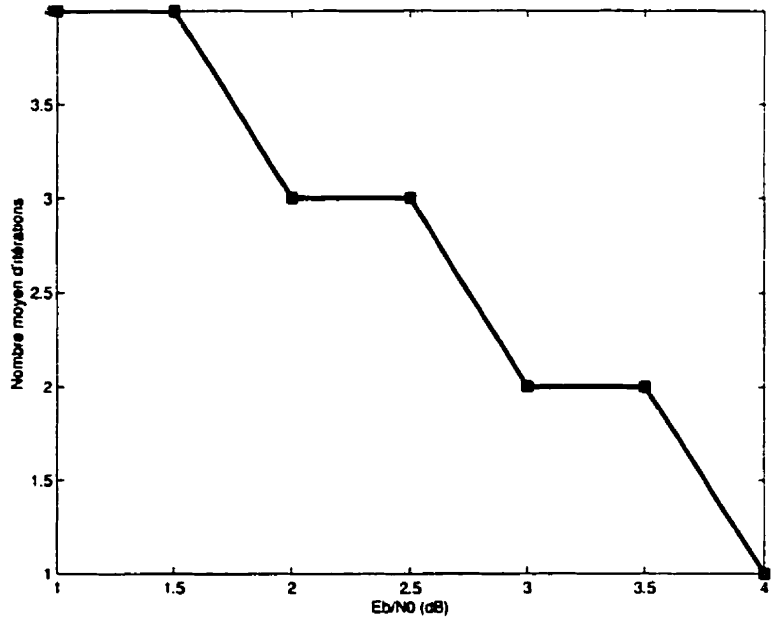


Figure 4.27: Nombre Moyen d'itérations du code turbo en mode parallèle utilisant le critère de la figure 4.26

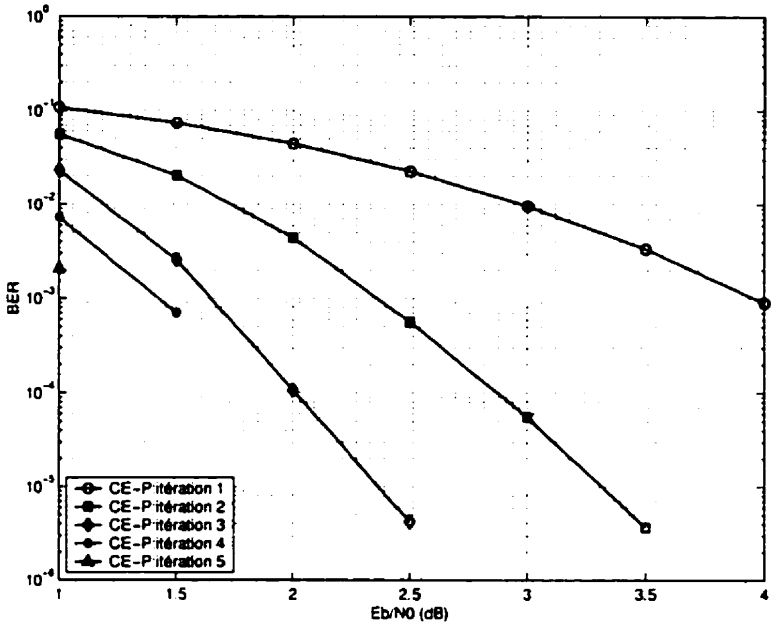


Figure 4.28: Critère d'arrêt CE-P: performance du code turbo en mode parallèle dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 1/3$ et taille d'entrelaceur aléatoire 1024

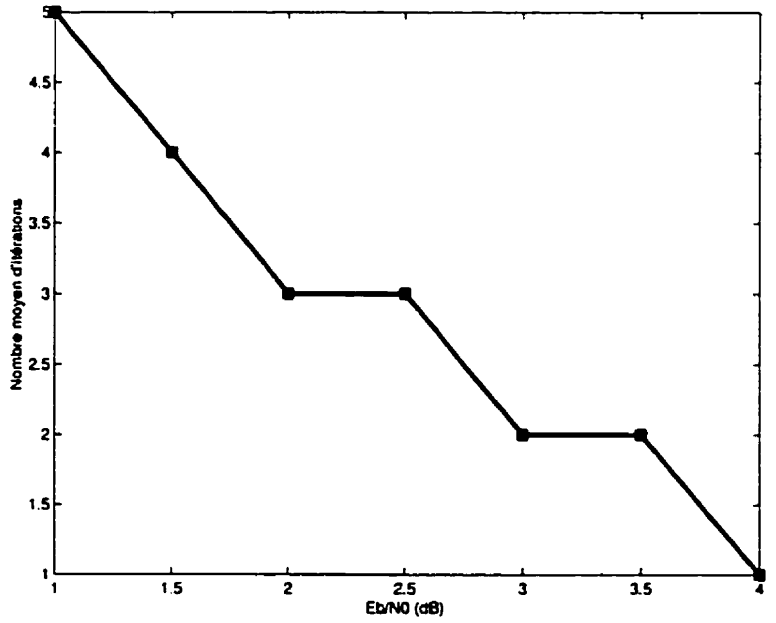


Figure 4.29: Nombre Moyen d'itérations du code turbo en mode parallèle utilisant le critère de la figure 4.28

Chapitre 5

Performance des codes Turbo perforés

5.1 Introduction

La perforation est le processus d'élimination systématique de certains symboles/ positions de la séquence codée à la sortie d'un encodeur. De cette façon, le rapport du nombre des bits d'information qui entrent dans le codeur et le nombre des symboles codés qui subsistent après la perforation, présente un taux résultant supérieur au taux de codage de l'encodeur non perforé [48].

La perforation a été appliquée dans le cas des codes turbo pour introduire moins de redondance et utiliser moins de largeur de bande que les codes dont le taux de codage est faible. Cependant et selon le patron de perforation, elle peut introduire

une dégradation de la performance de codes turbo suivant le choix de l'entrelaceur et des codeurs CRS. Cela est dû au manque de protection des bits d'information et à la diminution des poids des mots de code quand les symboles de parité sont éliminés périodiquement.

Dans ce chapitre, nous allons présenter l'importance du choix du patron de perforation et examiner ses effets sur la performance des codes turbo. Nous allons présenter une nouvelle classe de patrons de perforation qui permettent d'avoir des performances proches de la limite de Shannon. Nous déterminerons, pour les codes turbo perforés, leurs performances en termes de distance et de probabilité d'erreur en fonction du rapport $\frac{E_b}{N_0}$.

5.2 Spécification d'un code turbo perforé

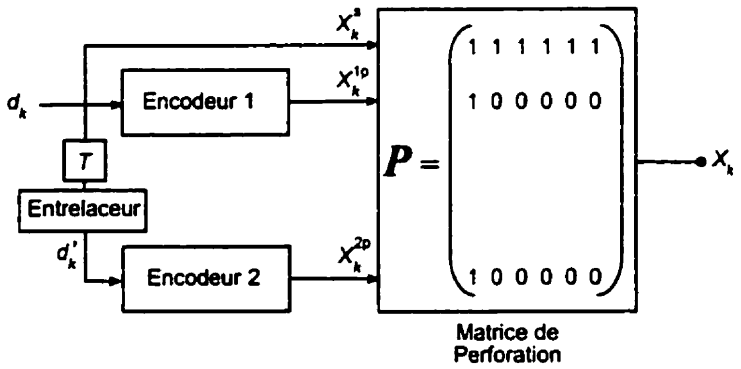


Figure 5.1: Encodeur turbo perforé à $R_{gp} = 6/8 = 3/4$

La figure 5.1, présente l'encodeur turbo vu précédemment. Sans perforation, le taux de codage R_{gp} est égal à 1/3. Pour chaque bit d_k, il existe trois sorties: X_k^s, X_k^{1p} et X_k^{2p} qui correspondent respectivement aux symbole systématique, symbole de parité 1 et symbole de parité 2. Afin d'atteindre des taux de codage plus élevés avec une meilleure performance, il faut perforer seulement les symboles de parité

et transmettre tous les symboles systématiques.

Nous avons exprimé le taux de codage du code turbo pour 2 codeurs CRS par:

$$R_{gp} = \frac{m}{m+2}, \quad m \geq 1 \tag{5.1}$$

cela signifie que pour chaque m bits d'information (période de perforation de longueur m), seulement 2 symboles de parité sont retenus. Autrement dit, pour chaque séquence de parité ($\mathbf{X}_p^1$ et $\mathbf{X}_p^2$) un symbole parmi m est transmis.

La performance du code turbo avec des taux élevés obtenue par perforation, dépend d'une part des caractéristiques de l'encodeur turbo d'origine (encodeur de taux faible) et d'autre part du patron de perforation qui est défini par une matrice binaire $\mathbf{P}$ appelée matrice de perforation.

Par conséquent, afin de trouver un bon code turbo perforé, on devrait prendre comme première étape un bon code d'origine (choix des polynômes générateurs et de la longueur de contrainte des codeurs) car généralement, les bons codes convolutionnels génèrent des bons codes turbo perforés. Cependant, cette partie a déjà fait l'objet de beaucoup de thèses et de mémoires de recherche. Ce qui fait nous ne traiterons pas cette partie mais nous utiliserons ce que les autres ont trouvé ([3],[44]).

Afin d'améliorer la performance des codes turbo perforés, la recherche revient alors à sélectionner les meilleurs patrons de perforation c'est-à-dire à la détermination de la matrice $\mathbf{P}$.

La matrice de perforation $\mathbf{P}$ sert à spécifier quels sont les symboles à transmettre. Sa dimension, pour deux codeurs CRS montés en parallèle, est trois lignes et m colonnes, où m est le numérateur du taux de codage.

Les éléments de la première ligne de la matrice **P** correspondent aux symboles systématiques. Ils sont tous égaux à 1 afin de ne pas être perforé. Quant à la deuxième et la troisième lignes, elles correspondent respectivement aux symboles de parité du codeur 1 et du codeur 2. Ainsi, dans une période de m symboles, les éléments de ces deux dernières lignes qui ont la valeur 1, spécifient les symboles de parité transmis alors que ceux ayant une valeur 0 représentent les symboles perforés.

Par exemple, considérons la matrice suivante qui peut être utilisée pour augmenter le taux de codage du code turbo de 1/3 à 4/6 c'est à dire 2/3:

$$\mathbf{P} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{5.2}$$

Notons par $P(k,l)$ ($1 \leq k \leq m$ et $1 \leq l \leq m$, m numérateur du taux de codage) les symboles de parité transmis, c'est-à-dire les éléments de la deuxième et troisième lignes de la matrice **P** qui ont la valeur 1 respectivement. Par exemple, la matrice dans (5.2) peut être représentée par la notation $P(1,1)$.

Le problème majeur de la perforation est le choix des bits à supprimer. En effet, les $k^{\text{ème}}$ et $l^{\text{ème}}$ positions des bits retenus respectivement de la première et la deuxième séquences de parité (les positions des 1 dans la deuxième et la troisième lignes de la matrice **P**), influent beaucoup sur la performance de décodage turbo.

Le nombre de patrons de perforation distincts croît avec le paramètre m. Cette croissance est d'autant plus rapide que le taux de codage résultant est grand. Ce nombre est égal à m^2. Par conséquent, la recherche de la meilleure matrice de perforation avec des rapports de taux de codage très élevés n'est pas une simple affaire.

Une des solutions pour trouver les meilleures positions des symboles retenus est

la détermination du spectre de poids du code turbo pour différentes combinaisons du patron de perforation. Ainsi, en utilisant l'approche dans [6] et en prenant les mêmes codes présentés dans [2], la matrice **P** qui fournit la meilleure performance en terme de probabilité d'erreur peut être trouvée. Cependant, pour certains codeurs CRS (à petite mémoire et faible distance libre) cette performance d'erreur reste encore trop faible. Pour résoudre ce problème, nous avons introduit des changements dans le patron de perforation. Dans la suite, nous désignerons patrons de perforation classiques, les patrons dont le nombre de colonnes de la matrice de perforation est l'entier m, numérateur de taux de codage.

5.3 Patron de perforation modifié

5.3.1 Motivation et problématique

Dans [20], Fan *et al* ont étudié la perforation des codes turbo avec une longueur de contrainte $K = 5$. Ils ont trouvé une nouvelle méthode de perforation pour les taux de codage $R_{gp} = 10/12$, $20/22$ et $30/32$. En effet, pour ceux-ci, quel que soit d'une part le patron de perforation classique et d'autre part le type et/ou la taille de l'entrelaceur, la performance des codes turbo reste très faible et loin de la capacité de Shannon.

Pour expliquer cette faible performance, Fan *et al* ont montré que le patron de perforation classique ne permet pas de sélectionner toutes les positions de la période de la réponse impulsionnelle du codeur CRS qui est $\leq 2^{K-1} - 1$. Le nombre de positions est égal à la période déjà mentionnée. Selon Fan, les positions de cette période sont équiprobables pour qu'elles soient sélectionnées, alors qu'avec le patron ordinaire seulement certaines de ces positions ont la possibilité d'être retenues tout le long de la séquence de parité.

Pour mieux expliquer le point de vue de Fan *et al*, on considère l'exemple du

tableau 5.1 (la case des positions dans la période 15 : contient chaque fois le reste de la division de la valeur de la position du symbole sélectionné par 15) qui montre les différentes positions sélectionnées dans une période 15 pour le taux de codage $R_{gp} = 10/12$. On remarque que seulement la $1^{ère}$, $6^{ème}$ et $11^{ème}$ positions sont retenues et les douze autres ne peuvent jamais être sélectionnées.

Tableau 5.1: Les positions de la période sélectionnées pour $R = 10/12$

positions des symboles sélectionnés	1	11	21	31	41	51	61	71	81	91
Position dans la période 15	1	11	6	1	11	6	1	11	6	1
positions des symboles sélectionnés	101	111	121	131	141	151	161	171	181	...
Position dans la période 15	11	6	1	11	6	1	11	6	1	...

Pour avoir toutes les positions de la période de la réponse impulsionnelle et augmenter ainsi la performance, Fan *et al* ont proposé de sélectionner pour chaque 50 symboles de parité successifs, le 1^{er}, le $12^{ème}$, le $23^{ème}$, le $34^{ème}$ et le $45^{ème}$ symboles comme l'indique le tableau 5.2.

Tableau 5.2: Nouvelle sélection pour $R_{gp} = 10/12$

positions des symboles sélectionnés	1	12	23	34	45	51	62	73	84	95
Position dans la période 15	1	12	8	4	15	6	2	13	9	5
positions des symboles sélectionnés	101	112	123	134	145	151	162	173	184	...
Position dans la période 15	11	7	3	14	10	1	12	8	4	...

Avec cette nouvelle technique de perforation (nouveau choix des bits retenus), la performance du code turbo est améliorée. De notre part et en se basant sur le même

Tableau 5.3: Autre façon de sélection pour $R_{gp} = 10/12$

positions des symboles sélectionnés	1	12	23	34	45	56	67	78	89	100
Position dans la période 15	1	12	8	4	15	11	7	3	14	10
positions des symboles sélectionnés	103	111	122	140	144	151	162	173	184	...
Position dans la période 15	13	6	2	5	9	1	12	8	4	...

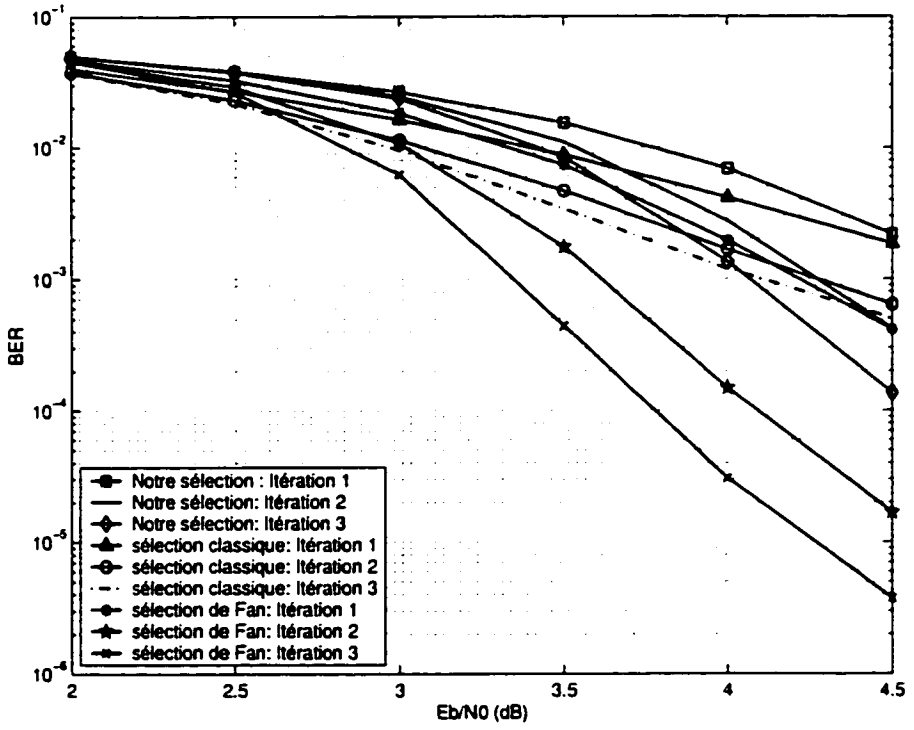


Figure 5.2: Performances des codes turbo perforés dans un canal AWGN avec $R_{gp} = 10/12$, $K = 5$, $G = (1, \frac{35}{23})$ et taille d'entrelaceur aléatoire 1024: Résultats de simulation correspondants aux tableaux 5.1, 5.2 et 5.3

Tableau 5.4: Différentes sélections des symboles de parité pour $R_{gp} = 6/8$

Patron classique(ancienne méthode)										
positions des symboles sélectionnés	1	7	13	19	25	31	37	43	49	55
Position dans la période 15	1	7	13	4	10	1	7	13	4	10
positions des symboles sélectionnés	61	67	73	79	85	91	97	103	109	...
Position dans la période 15	1	7	13	4	10	1	7	13	4	...
Patron Modifié (nouvelle méthode)										
positions des symboles sélectionnés	1	7	13	19	25	32	38	44	50	56
Position dans la période 15	1	7	13	4	10	2	8	14	5	11
positions des symboles sélectionnés	63	69	75	81	87	91	97	103	109	...
Position dans la période 15	3	9	15	6	12	1	7	13	4	...

principe que Fan *et al*, nous avons choisi d'autres façons de sélectionner les symboles où toutes les positions sont retenues comme montré au tableau 5.3. Malheureusement les simulations illustrées à la figure 5.2 n'ont montré aucune amélioration de la performance par rapport à l'utilisation des patrons de perforation classiques.

En se basant sur la méthode de Fan *et al* qui consiste à sélectionner toutes les positions de la période de la réponse impulsionnelle, nous avons aussi essayé de déterminer la performance du code turbo pour le taux de codage $R_{gp} = 6/8$ avec une longueur de contrainte $K = 5$ et $G = (1, \frac{31}{23})$. Le tableau 5.4 indique les positions retenues en utilisant l'ancienne et la nouvelle méthodes. Les résultats de simulation correspondants à ce tableau sont illustrés à la figure 5.3, montrant que la sélection de toutes les positions ne donne pas forcément une meilleure performance, au contraire elle peut faire chuter celle-ci. Par conséquent, l'explication proposée

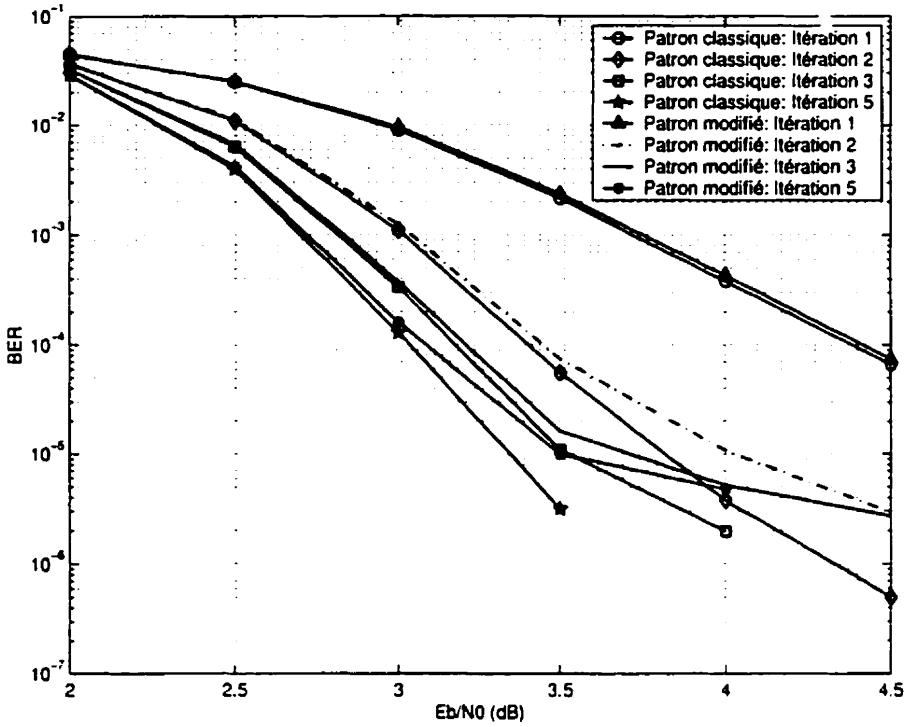


Figure 5.3: Comparaison entre l'ancienne et la nouvelle méthodes de perforation pour un taux de codage $R_{gp} = 6/8$ avec $K = 5$, $G = (1, \frac{31}{23})$ et taille d'entrelaceur aléatoire 900

par Fan *et al* ne semble pas être toujours valide.

Nous avons trouvé un autre contre exemple pour le taux de codage $R_{gp} = 12/14$. Le tableau 5.5 nous montre les positions prises dans la période 15 de la réponse impulsionnelle avec $K = 5$ et $G = (1, \frac{31}{23})$, en utilisant l'ancien patron de perforation et un autre patron de perforation où toutes les positions de la période de la réponse impulsionnelle sont sélectionnées.

Les résultats de simulation selon ce tableau (5.5) sont montrés à la figure 5.4. Il est clair dans ce cas aussi que la sélection de toutes les positions de la période de la réponse impulsionnelle (méthode de Fan), n'est pas l'explication à l'amélioration

Tableau 5.5: Différentes sélections des symboles de parité pour $R_{gp} = 12/14$

Patron classique(ancienne méthode)										
positions des symboles sélectionnés	1	13	25	37	49	61	73	85	97	109
Position dans la période 15	1	13	10	7	4	1	13	10	7	4
positions des symboles sélectionnés	121	133	145	157	169	181	193	205	217	...
Position dans la période 15	1	13	10	7	4	1	13	10	7	...
Patron Modifié (nouvelle méthode)										
positions des symboles sélectionnés	1	13	25	37	49	62	74	86	98	110
Position dans la période 15	1	13	10	7	4	2	14	11	8	5
positions des symboles sélectionnés	123	135	147	159	170	181	193	205	217	...
Position dans la période 15	3	15	12	9	6	1	13	10	7	...

des performances des codes turbo.

Alors, comment expliquons-nous ces phénomènes et éventuellement, pouvons nous généraliser ceci pour tous les codeurs CRS constituant les codes turbo pour différentes longueurs de contraintes?

5.3.2 Généralisation de la nouvelle classe de patron de perforation

Dans un premier temps, nous avons cherché à déterminer le meilleur patron de perforation pour un code turbo perforé de taux de codage $R_{gp} = 6/8$ avec des codeurs CRS de $K = 3$ c'est-à-dire $M = 2$. Nous avons testé toutes les combinaisons pour obtenir $P(5, 2)$ qui donne les meilleures positions des symboles de

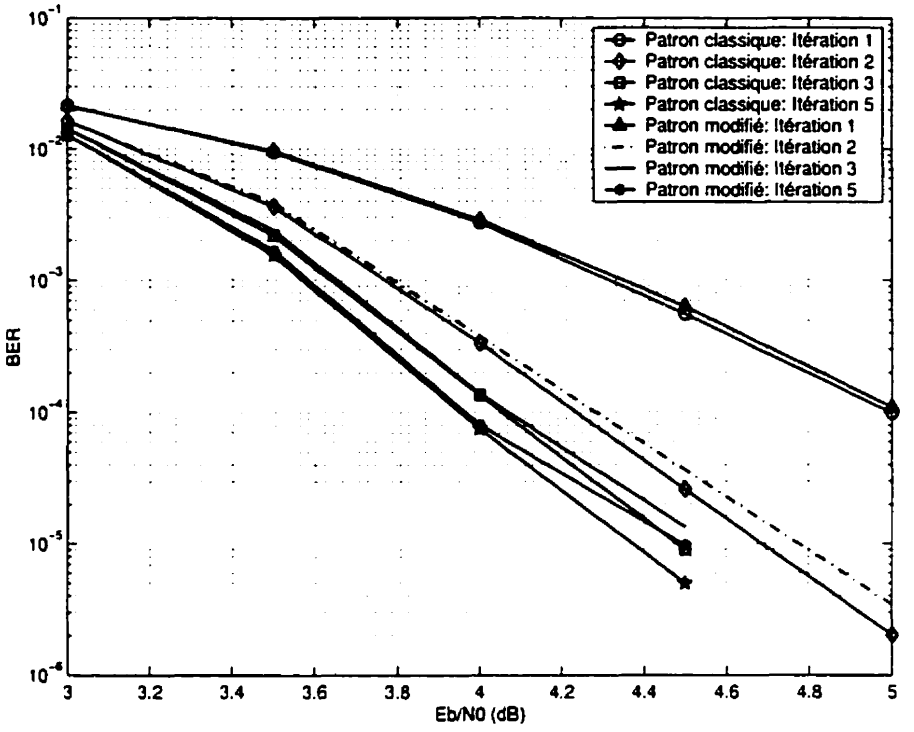


Figure 5.4: Comparaison entre l'ancienne et la nouvelle méthodes de perforation pour un taux de codage $R_{gp} = 12/14$ dans un canal AWGN avec $K = 5$, $G = (1, \frac{31}{23})$, taille d'entrelaceur aléatoire 1024

parité. Ensuite et à partir de la figure 5.5, nous avons remarqué que même avec une augmentation de la taille de l'entrelaceur, la performance en terme de probabilité d'erreur reste inchangée et très faible. Pour augmenter et améliorer cette performance une nouvelle classe généralisée des patrons de perforation a été développée.

En effet, nous avons trouvé des relations de dépendance, d'une part entre la longueur de contrainte $K = M + 1$ et le taux de codage défini dans (5.1), et d'autre part entre la longueur L_I de la réponse impulsionnelle du codeur élémentaire CRS et le numérateur du taux de codage global.

Si le numérateur m ou le dénominateur $m + 2$ du taux de codage est un multi-

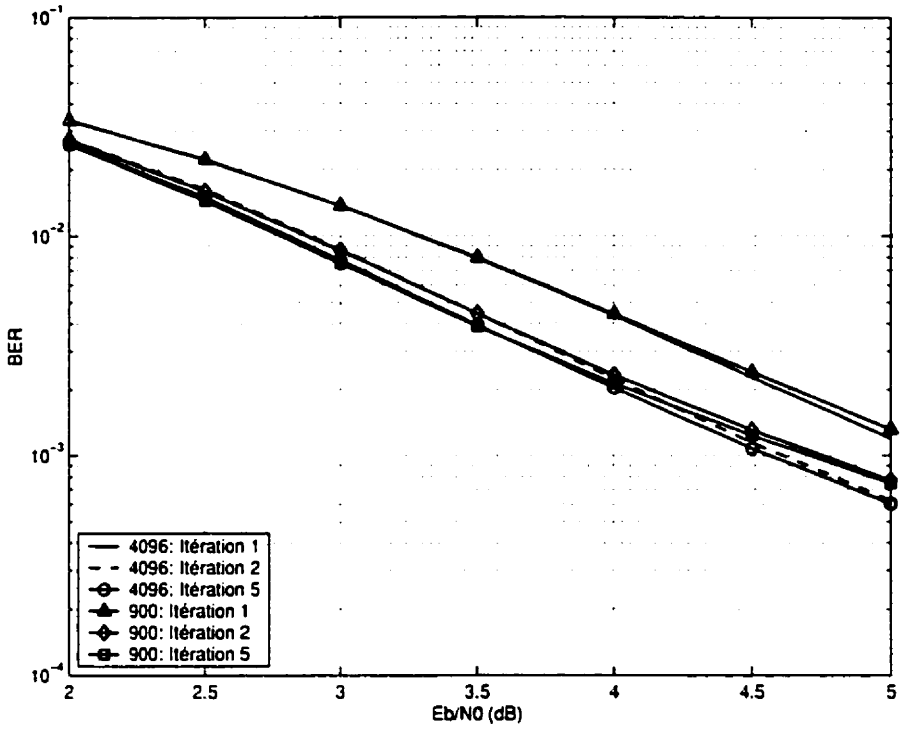


Figure 5.5: Performance des codes turbo utilisant un patron de perforation classique dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, $R_{gp} = 6/8$ et tailles d'entrelaceur aléatoire 900 et 4096

ple de K (pour $m \geq K$) et/ou si m est un multiple de L_I (pour $m \geq L_I$), alors la matrice de perforation peut avoir $K \times m$ colonnes sous certaines conditions. Dans ce cas, il est clair que pour chaque $K \times m$ bits d'information, K symboles de parité 1 et K symboles de parité 2 seulement sont retenus. Cependant, le choix des positions de ces $2K$ symboles est particulier. Pour les m premiers symboles, on choisit $P(1, 1)$. Par contre, pour la série des m symboles suivants, il faut choisir une autre matrice caractérisée par $P(2, 2)$. On procède ainsi de suite jusqu'aux les m derniers symboles, pour lesquels nous prenons la notation $P(K, K)$.

Le tableau 5.6 montre la longueur L_I de la réponse impulsionnelle de différents codeurs CRS.

Tableau 5.6: Longueur L_I de la réponse impulsionnelle de différents codeurs CRS de taux $R = 1/2$

K	G_1	G_2	L_I
3	7	5	3
4	15	17	7
4	17	15	4
5	23	35	15
5	37	21	5
5	23	31	15

Une simplification de notre raisonnement est de considérer un exemple afin de déterminer la matrice de perforation du code turbo avec un taux de codage $R_{gp} = 6/8$ et une longueur de contrainte des codes CRS élémentaires $K = 3$ et $R = 1/2$. En effet, comme l'indique (5.3), la matrice $\mathbf{P}$ est équivalente à une combinaison périodique de trois autres caractérisées chacune par la notation respective $P(1,1)$, $P(2,2)$ et $P(3,3)$. Autrement dit, au lieu d'utiliser une seule matrice fixe avec un nombre de colonnes m pour tous les bits d'information transmis, nous utilisons dans chaque période de $K \times m$ bits, K différentes matrices, chacune de taille $3 \times m$.

$$\mathbf{P} = \left(\begin{array}{cccccc|cccccc|cccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right) \tag{5.3}$$

Les résultats de simulation du code turbo avec le taux de codage $R_{gp} = 6/8$ utilisant les patrons classique et modifié sont illustrés à la figure 5.6. Nous pouvons avoir un gain de codage de 2 dB avec la nouvelle matrice de perforation.

D'un autre côté, avec la nouvelle méthode de perforation, nous trouvons dans (5.4), le nouveau patron $\mathbf{P}$ d'un code turbo avec un taux de codage $R_{gp} = 4/6$ et une longueur de contrainte des codes CRS élémentaires $K = 3$.

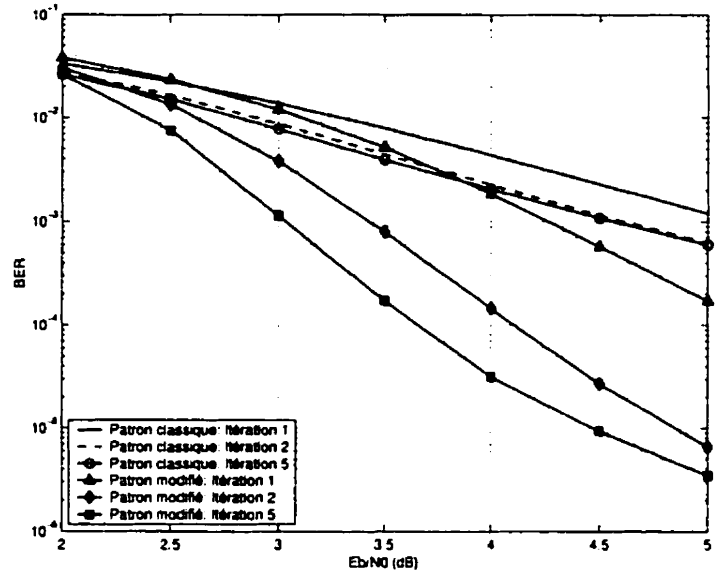


Figure 5.6: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, $R_{gp} = 6/8$ et taille d'entrelaceur aléatoire 900

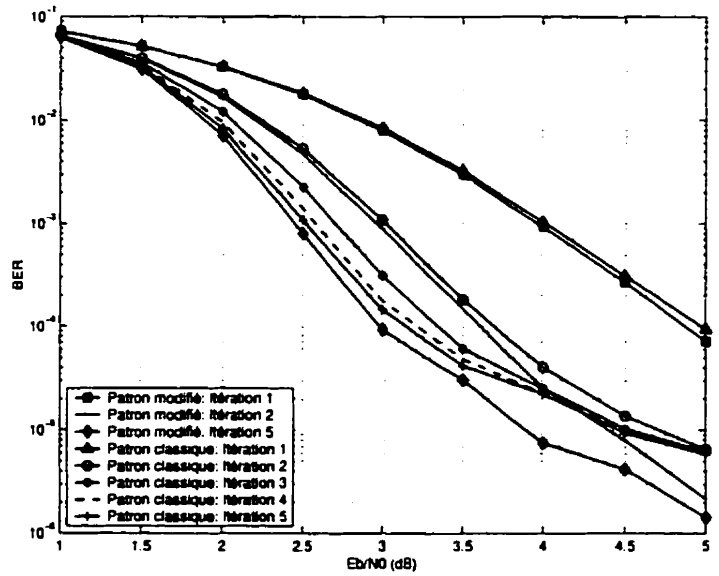


Figure 5.7: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, $R_{gp} = 4/6$ et taille d'entrelaceur aléatoire 900

$$\mathbf{P} = \left(\begin{array}{cccc|cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right) \tag{5.4}$$

La figure 5.7 montre les résultats de simulation du code turbo de taux de codage $R_{gp} = 4/6$ avec l'ancien et le nouveau patrons de perforation. Le gain de codage dans ce cas est de l'ordre de 1 dB.

Pour tous les taux de codage qui ne satisfont pas l'une et/ou l'autre condition de multiplicité, nous devons utiliser des simples matrices de perforation (matrices classiques) qui fournissent les meilleures performances pour les codes turbo. Dans ce cas et pour chacun de ces taux de codage, le problème serait alors de trouver les meilleures positions des symboles de parité retenus, c'est-à-dire le meilleur $P(k, l)$ ($1 \leq k \leq m$ et $1 \leq l \leq m$, m numérateur du taux de codage).

5.4 Analyse de l'effet du nouveau patron de perforation

Considérons un code turbo de taux de codage $R_{gp} = 3/5$ avec deux codeurs élémentaire CRS de longueur de contrainte $K = 3$ et $R = 1/2$. Un exemple de l'ancienne matrice de perforation **P1** est donné par:

$$\mathbf{P1} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \tag{5.5}$$

Par contre, la nouvelle matrice de perforation **P2** est définie par:

$$\mathbf{P2} = \left(\begin{array}{ccc|ccc|ccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right) \tag{5.6}$$

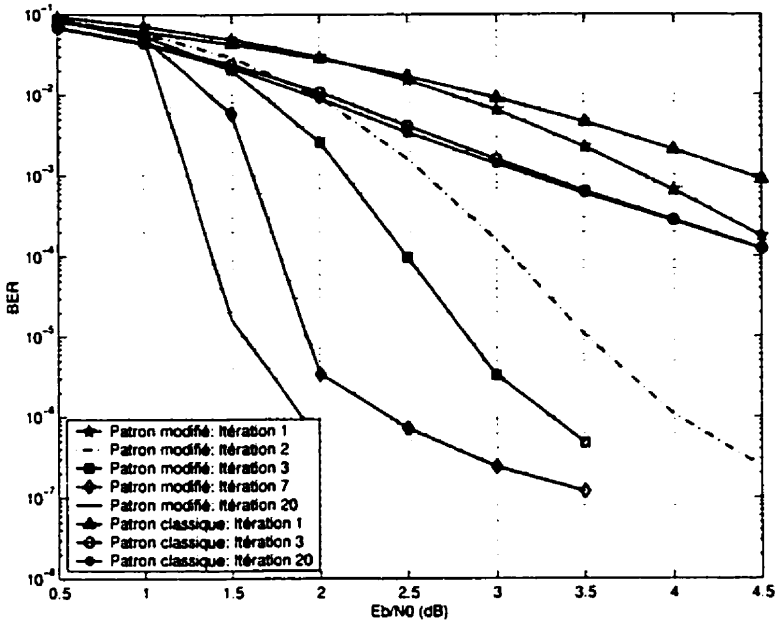


Figure 5.8: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, $R_{gp} = 3/5$ et taille d'entrelaceur aléatoire 65536

Les résultats de simulation du code turbo perforé utilisant les patrons **P1** et **P2** sont illustrés à la figures 5.8.

La figure 5.9 illustre le treillis du codeur CRS sans perforation. Afin d'avoir un code turbo de taux de codage $R_{gp} = 3/5$, les treillis de ces codeurs CRS doivent être modifiés en fonction du patron de perforation. La figure 5.10 montre le treillis de l'un des deux codeurs élémentaires de taux de codage résultant $R = 3/4$ suivant la matrice **P1**. Il est clair, qu'avec cette matrice de perforation traditionnelle, le treillis de ces codes est fixe pour tous les bits d'information en entrée. Alors qu'avec la matrice modifiée, ce treillis change périodiquement K fois. La figure 5.11 illustre le nouveau treillis du codeur CRS après l'utilisation du nouveau patron de perforation. Chacune de ces profondeurs correspond à une des K matrices de taille $3 \times m$.

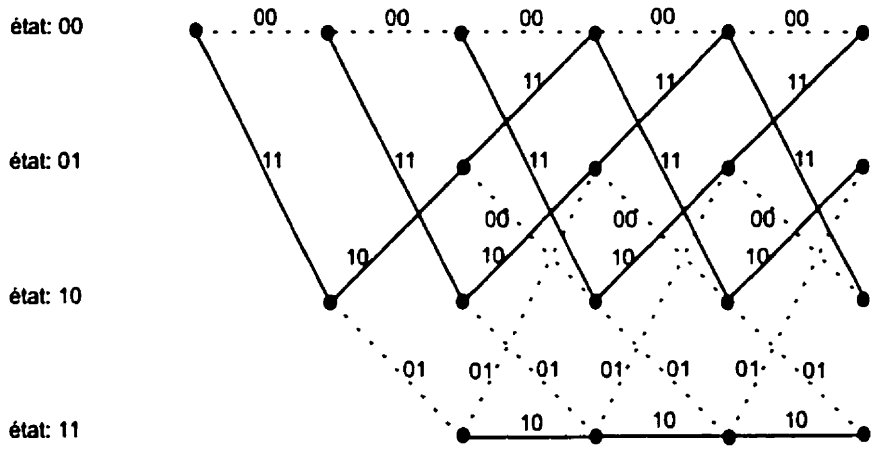
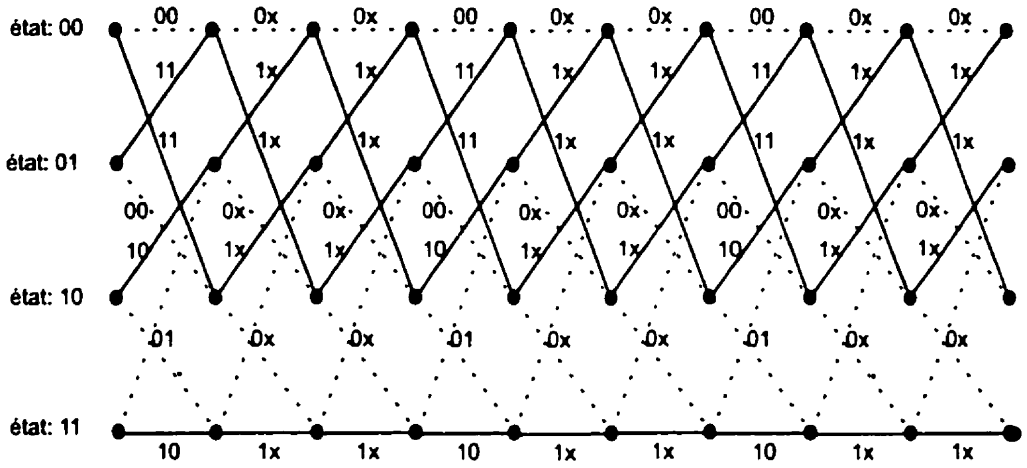


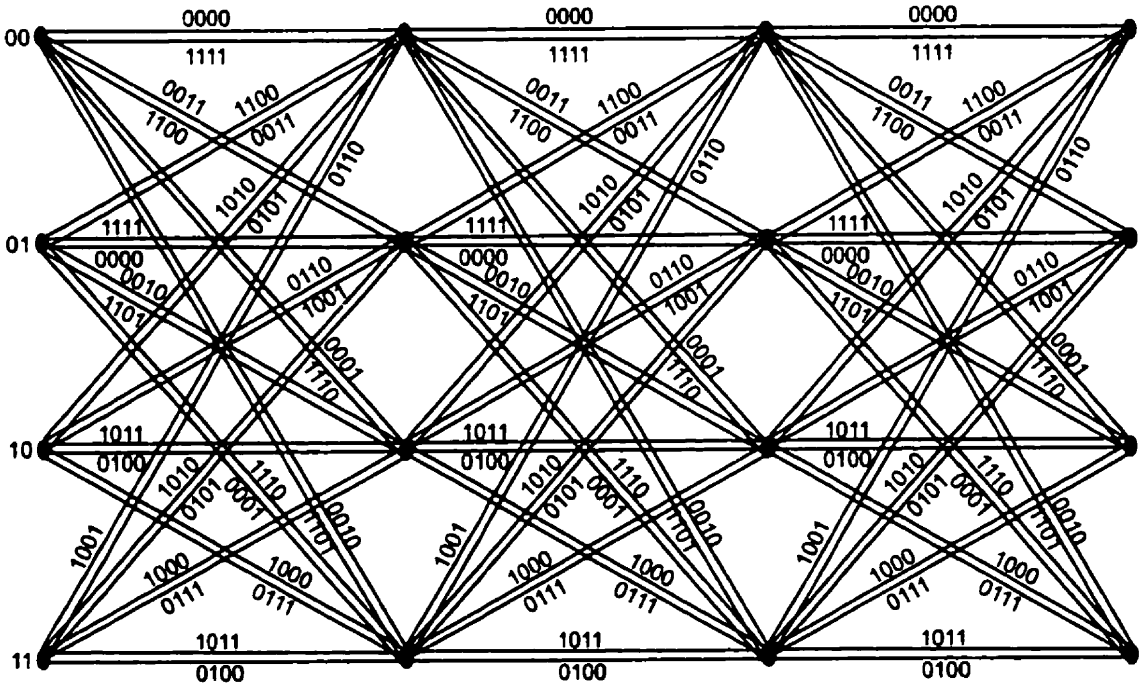
Figure 5.9: Treillis du code élémentaire CRS de la figure 1.8 avec un taux de codage $R = 1/2$

En fait, les K changements de treillis est en relation avec la longueur de contrainte des codeurs élémentaires identiques CRS. Ainsi, d'une itération à une autre et avec l'addition de l'effet de l'entrelaceur, l'utilisation de la nouvelle matrice de perforation par le codeur turbo permet de décorréler davantage les séquences d'information extrinsèque $\mathbf{Le}^1$ et $\mathbf{Le}^2$ avec les différentes entrées respectives de DEC2 et DEC1. Par contre, en utilisant l'ancien patron de perforation, la corrélation entre les différentes entrées d'un tel décodeur (DEC1 et/ou DEC2) augmente et par la suite les codes turbo saturent très vite avec des faible performances.

D'autre part, grâce à la bonne élimination périodique des mots de code à faible poids, le nouveau patron permet d'augmenter la distance libre d_{free} du code turbo par rapport à l'ancienne matrice de perforation. Autrement dit, une bonne élimination des symboles de parité qui contribuent plus à augmenter le poids des mots de code. Par conséquent, la nouvelle matrice de perforation améliore plus les performances asymptotiques des codes turbo (les codes turbo saturent moins vite). Les tableaux 5.7 et 5.8 montrent le spectre du code turbo utilisant la nouvelle et l'ancienne matrices de perforation pour un taux de codage $R_{gp} = 6/8$ et de

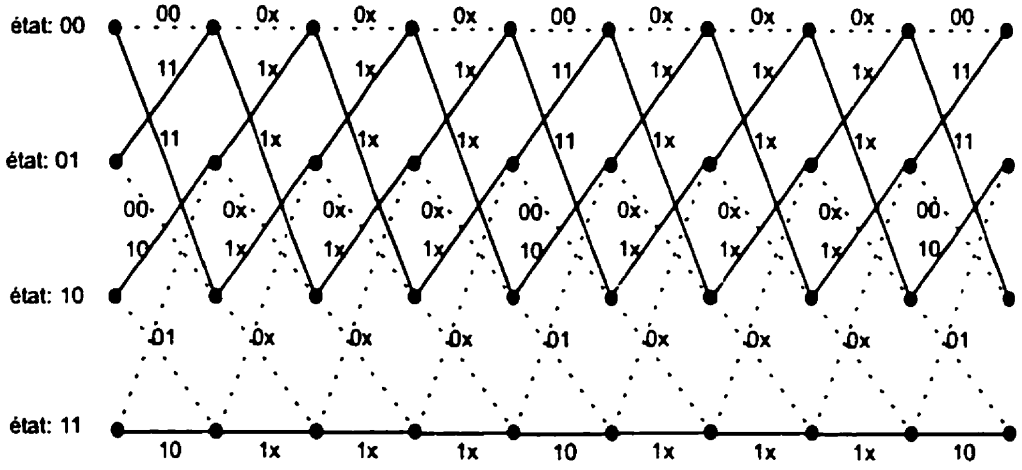


a) treillis du codeur CRS perforé utilisé par l'algorithme Log-MAP
x: symbole perforé

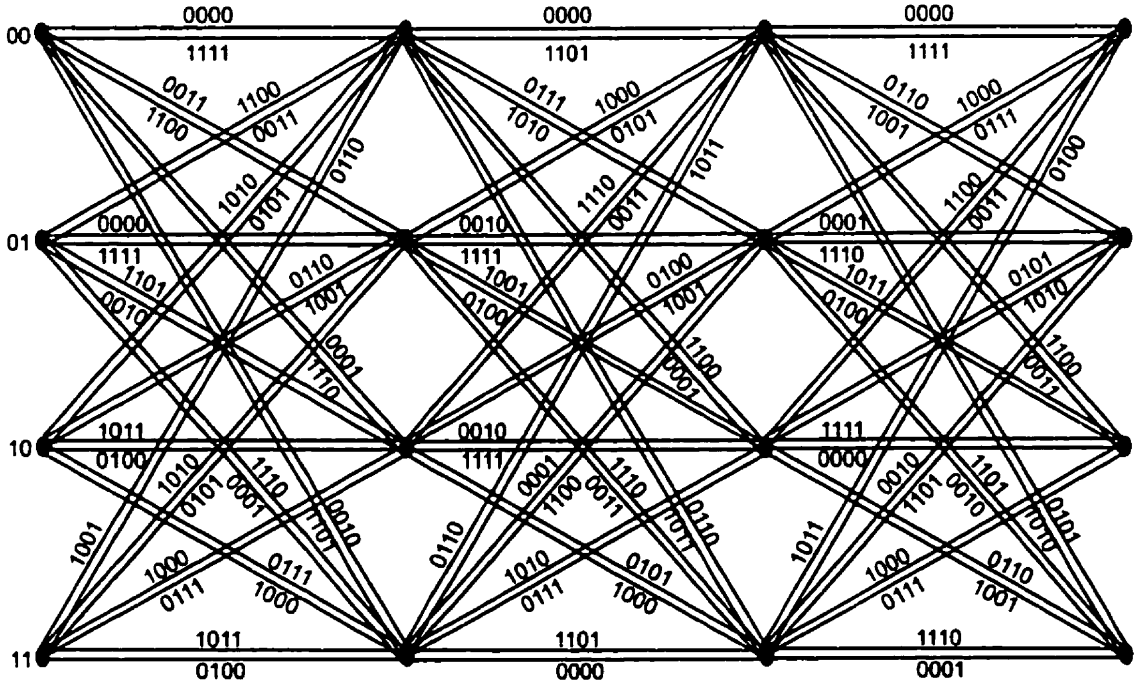


b) treillis du codeur CRS perforé équivalent

Figure 5.10: Treillis perforé du code élémentaire CRS de la figure 5.9 suivant la matrice de perforation **P1** avec $R = 3/4$



a) treillis du codeur CRS perforé utilisé par l'algorithme Log-MAP
x: symbole perforé



b) treillis du codeur CRS perforé équivalent

Figure 5.11: Treillis perforé du code élémentaire CRS de la figure 5.9 suivant la matrice de perforation **P2** avec $R = 3/4$

Tableau 5.7: Spectre de poids des codes turbo avec $R_{gp} = 6/8$, longueur de l'entrelaceur 18

Code élémentaire CRS d'origine				Codes turbo perforés		
M	G_1	G_2	d_f	P	d_f	$(A_{w,d}^{C_p}, \quad d = d_f, d_f + 1, \ldots)$
2	7	5	5	classique	4	(2, 0, 14, 85, 454, 1306, 2878, 5819, 9592, 12716, 13674, 12123, 9304)
2	7	5	5	modifié	5	(2, 19, 117, 420, 1189, 2973, 5855, 9440, 12903, 13959, 11995, 8908)

Tableau 5.8: Spectre de poids des codes turbo avec $R_{gp} = 6/8$, longueur de l'entrelaceur 50

Code élémentaire CRS d'origine				Codes turbo perforés		
M	G_1	G_2	d_f	P	d_f	$(A_{w,d}^{C_p}, \quad d = d_f, d_f + 1, \ldots)$
2	5	7	5	classique	4	(6, 12, 164, 988, 5062, 149326, 131902, 608936, 3547856, 21426550, 86964588)
2	5	7	5	modifié	6	(18, 110, 1308, 9238, 61286, 379002, 2893682, 21048260, 109011038)

tailles d'entrelaceur respectives 18 et 50. Ainsi, leurs bornes union respectives sont illustrées sur les figures 5.12 et 5.13.

5.5 Résultats de simulation

Il est bien connu que pour une même longueur de contrainte avec le même vecteur générateur, la perforation entraîne une dégradation de la performance des codes turbo proportionnellement au taux de codage voulu. Dans cette section, nous n'allons pas encore détailler ce phénomène qui semble logique étant donné la diminution de la distance libre, mais nous traiterons plutôt les effets combinés de

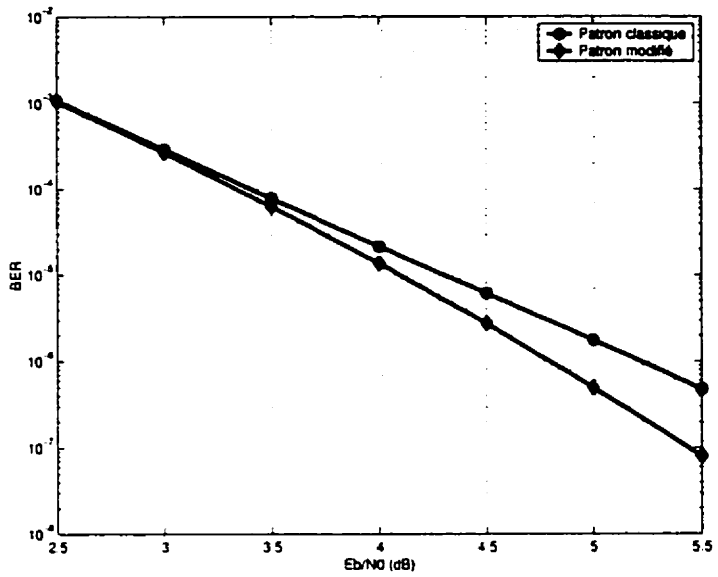


Figure 5.12: Borne union du code turbo utilisant l'ancien et le nouveau patrons de perforation pour un taux de codage $R = 6/8$ et taille d'entrelaceur aléatoire: 18

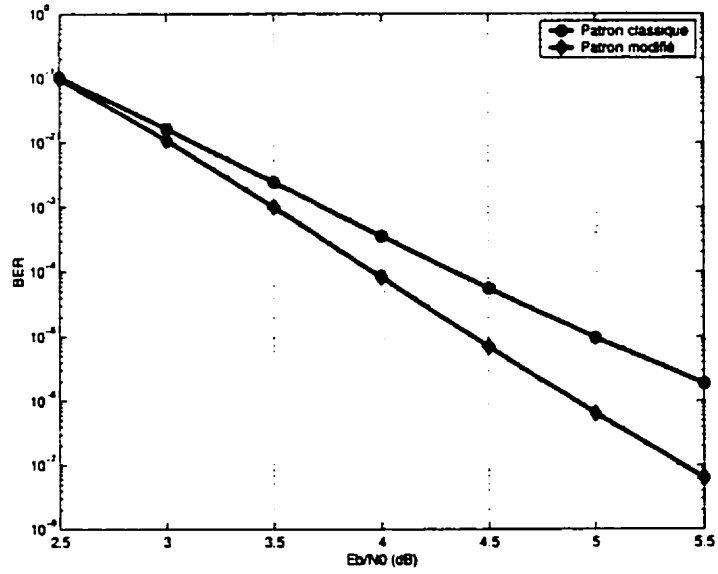


Figure 5.13: Borne union du code turbo utilisant l'ancien et le nouveau patrons de perforation pour un taux de codage $R = 6/8$ et taille d'entrelaceur aléatoire: 50

la perforation avec la taille des entrelaceurs, le nombre d'itérations et le choix du codeur élémentaire CRS sur la performance des codes turbo.

Par ailleurs, un grand nombre des résultats de simulation pour différentes valeurs de K est présenté dans l'annexe IV.

5.5.1 Effet combiné de la perforation et de la taille des entrelaceurs sur la performance des codes turbo

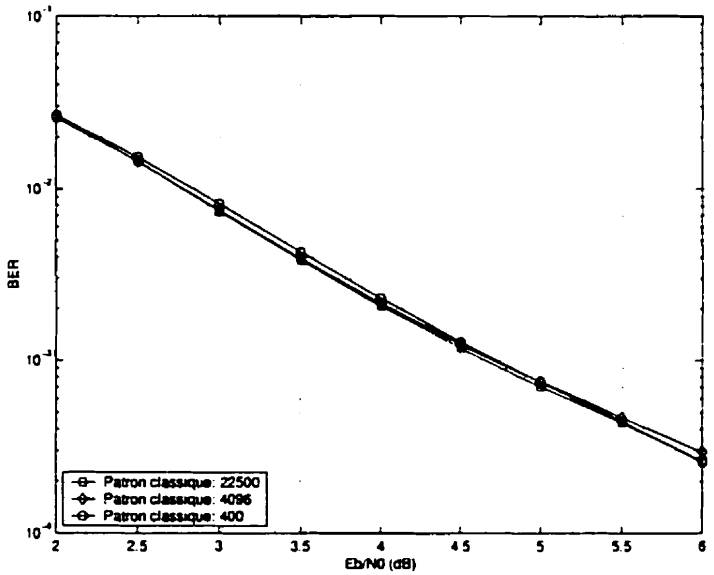


Figure 5.14: Influence de la taille de l'entrelaceur aléatoire sur la performance des codes turbo perforés avec le patron classique: 5ème itération, $K = 3$, $G = (1, 5/7)$, canal AWGN, $R_{gp} = 6/8$

Les figures 5.14 et 5.15 illustrent les résultats de simulation d'un codeur turbo utilisant respectivement l'ancienne et la nouvelle matrices de perforation avec une longueur de contrainte des codeurs élémentaires $K = 3$, $G = (1, 5/7)$ et $R_{gp} = 3/5$. Ainsi, une augmentation de la taille des entrelaceurs ne permet pas d'améliorer les performances d'un codeur turbo perforé utilisant un patron classique. Par contre,

avec le patron modifié, le comportement du code turbo est normal: plus la taille des entrelaceurs est grande, meilleures sont les performances.

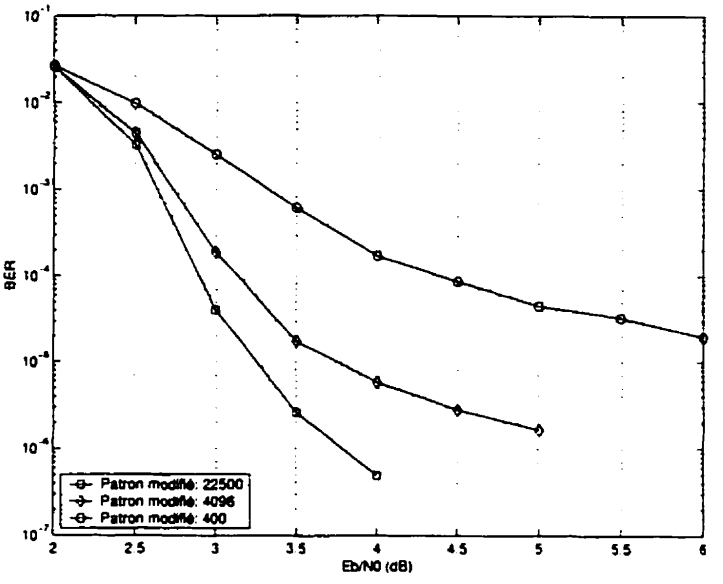


Figure 5.15: Influence de la taille de l'entrelaceur aléatoire sur la performance des codes turbo perforés avec le patron modifié: 5ème itération, $K = 3$, $G = (1, 5/7)$, canal AWGN, $R_{gp} = 6/8$

5.5.2 Effet combiné de la perforation et le choix des codeurs CRS sur la performance des codes turbo

La figure 5.16 illustre les performances obtenues avec les codes turbo perforés du taux de codage $R_{gp} = 6/8$, pour des vecteurs générateurs $G = (1, 7/5)$ et $G = (1, 5/7)$. La longueur de contrainte des codeurs CRS élémentaires est $K = 3$. Nous remarquons que le vecteur $G = (1, 7/5)$ fournit un gain de codage de l'ordre de 1.25 dB par rapport à $G = (1, 5/7)$.

Pour le cas d'un codeur CRS avec $K = 4$ et $G = (1, 17/15)$ et d'après la figure 5.17, nous constatons que le gain de codage entre les patrons de perforation

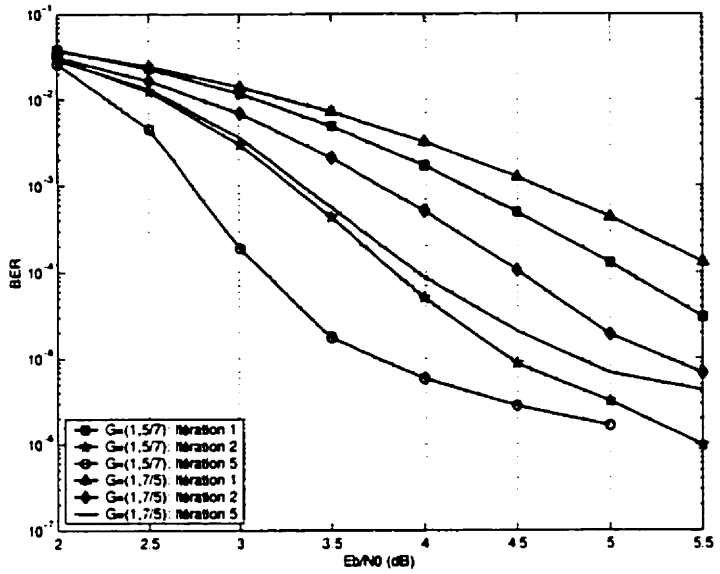


Figure 5.16: Influence du choix des codes élémentaires sur la performance des codes turbo perforés avec le patron modifié dans un canal AWGN avec $K = 3$, $R_{gp} = 6/8$ et taille de l'entrelaceur aléatoire 4096

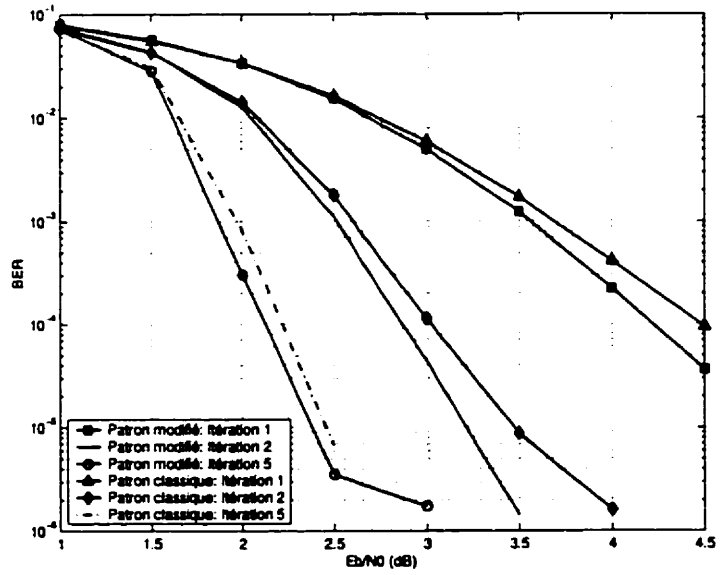


Figure 5.17: Performances des codes turbo perforés utilisant les patrons de perforation classique et modifié avec $K = 4$, $G = (1, 17/15)$, canal AWGN, $R_{gp} = 4/6$ et taille d'entrelaceur aléatoire 3600

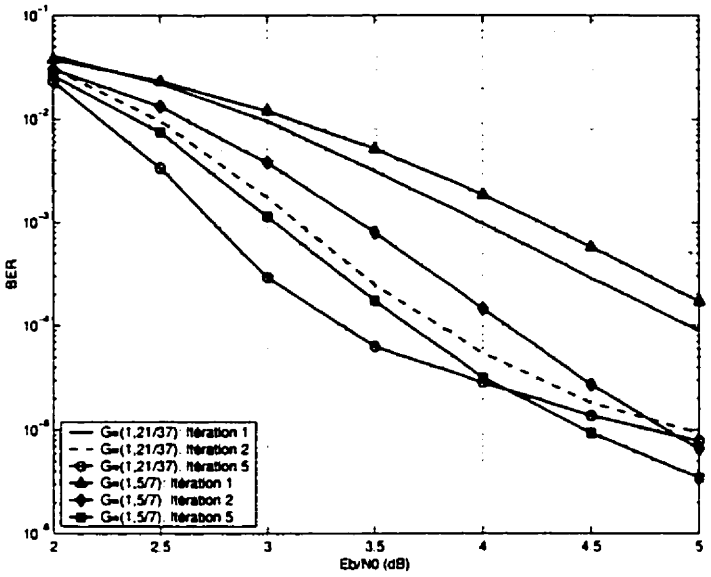


Figure 5.18: Performances des codes turbo perforés utilisant les patrons de perforation classique et modifié respectivement pour $K = 5$, $G = (1, 21/37)$ et $K = 3$, $G = (1, 5/7)$ dans canal AWGN avec $R_{gp} = 6/8$ et taille d'entrelaceur aléatoire 900

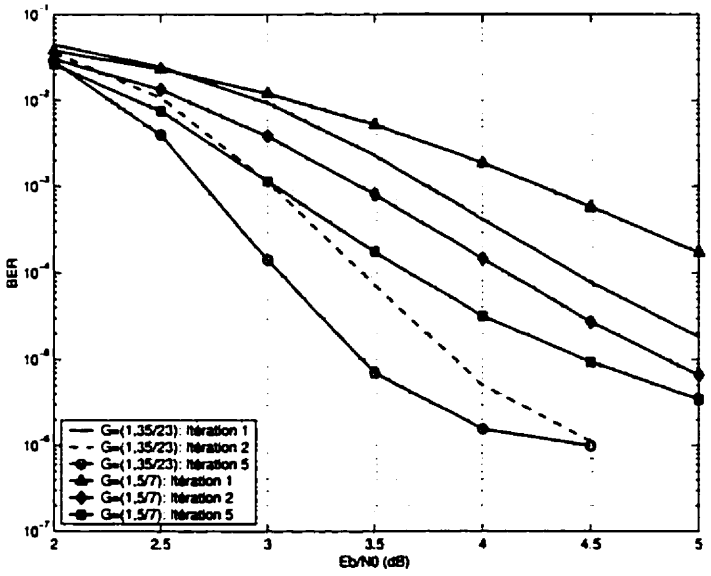


Figure 5.19: Performances des codes turbo perforés utilisant les patrons de perforation classique et modifié respectivement pour $K = 5$, $G = (1, 35/23)$ et $K = 3$, $G = (1, 5/7)$ dans canal AWGN avec $R_{gp} = 6/8$ et taille d'entrelaceur aléatoire 900

classique et modifié n'est pas élevé lorsque m est un multiple de K. Par contre avec $G = (1, 15/17)$, la performance des codes turbo s'améliore énormément et se rapproche de celle obtenue avec $G = (1, 17/15)$. Par conséquent, dans l'annexe IV, nous présenterons pour $K = 4$ et $G = (1, 17/15)$ seulement les résultats de simulation des codes turbo perforés utilisant le patron de perforation modifié telle que la valeur de m ou $m + 2$ est un multiple de K. Pour $G = (1, 17/15)$, dans la même annexe, les résultats de simulation sont présentés seulement pour m multiple de L_I.

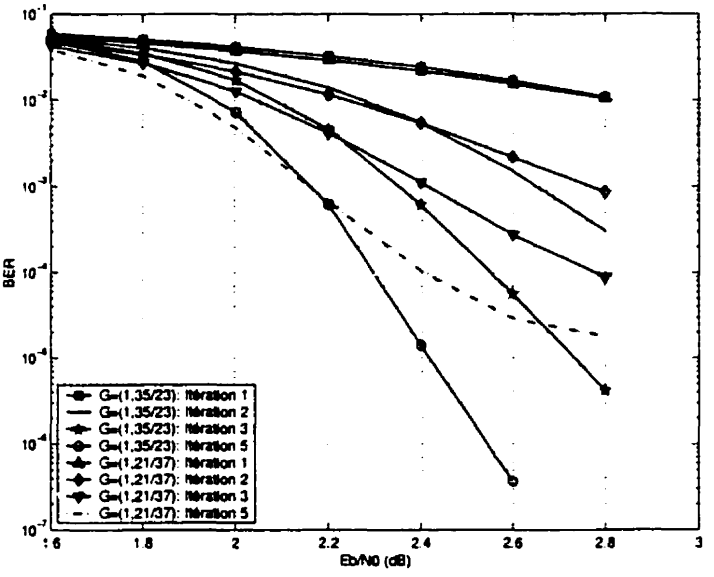


Figure 5.20: Performance des codes turbo avec le patron de perforation modifié dans un canal AWGN avec $K = 5$, $R_{gp} = 5/7$ et taille d'entrelaceur aléatoire 4096

La figure 5.18, montre les résultats de simulation du code turbo de taux de codage $R_{gp} = 6/8$ utilisant les patrons de perforation classique et modifié respectivement pour $K = 5$ avec $G = (1, 21/37)$ et $K = 3$ avec $G = (1, 5/7)$. Nous remarquons que pour le même nombre d'itérations (5 par exemple) et à partir de $\frac{E_b}{N_0} = 4$ dB, il est préférable d'utiliser un codeur CRS avec une longueur de contrainte $K = 3$ dans un code turbo car d'une part les performances sont meilleures

et d'autre part la complexité est plus faible. Toutefois, pour une meilleure performance en termes de probabilités d'erreur par bit, il faudrait choisir le codeur CRS avec $K = 5$ et $G = (1, 35/23)$, que se soit pour un taux de codage $R_{gp} = 6/8$ avec un patron classique ou pour $R_{gp} = 10/12$ avec une matrice de perforation modifiée comme l'indiquent les figures respectives 5.19 et 5.20.

5.5.3 Effet combiné de la perforation et du nombre d'itérations sur la performance des codes turbo

Dans le cas du codeur turbo perforé avec une matrice de perforation classique comme l'indique la figure 5.21, l'augmentation du nombre d'itérations n'introduit plus d'amélioration de la performance. Au delà de 3 itérations, le codeur turbo converge et il serait inutile de continuer le processus de décodage.

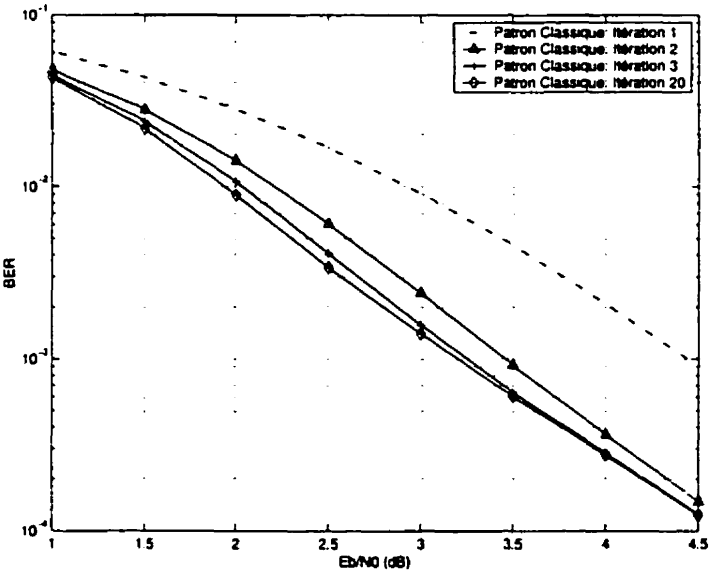


Figure 5.21: Influence du nombre d'itérations sur la performance des codes turbo perforés avec le patron classique dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, canal AWGN, $R_{gp} = 3/5$ et taille d'entrelaceur aléatoire 65536

Cependant, avec le nouveau patron de perforation, les performances du code

turbo perforé retrouvent leurs comportements habituels, c'est-à-dire, plus le nombre d'itérations est élevé, plus la probabilité d'erreur par bit diminue. Ce qui illustre l'explication déjà mentionnée au paragraphe précédent, concernant l'augmentation de la décorrélation entre les entrées d'un tel décodeur lors d'utilisation d'un treillis variable. En d'autres termes plus la corrélation est faible plus la saturation du code turbo est retardée.

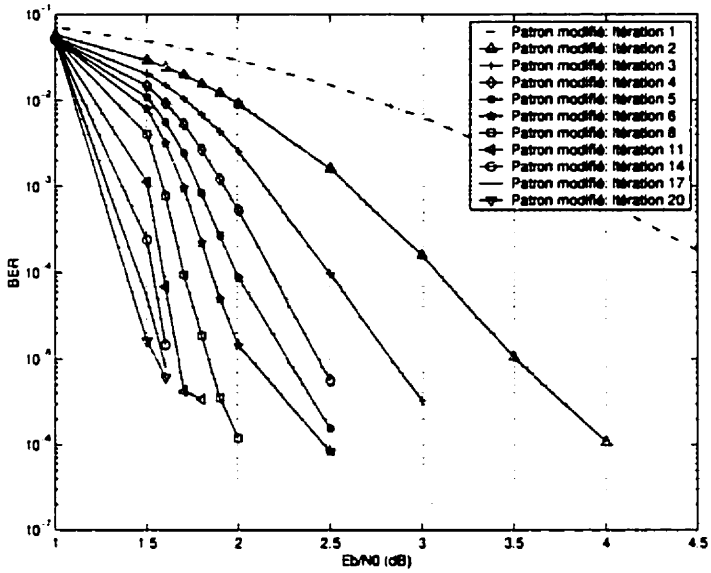


Figure 5.22: Influence du nombre d'itérations sur la performance des codes turbo perforés avec le patron modifié dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, canal AWGN, $R_{gp} = 3/5$ et taille d'entrelaceur aléatoire 65536

5.6 Conclusion

Dans ce chapitre, nous avons présenté les codes turbo perforés pour des taux de codage $> 1/2$. L'effet de la perforation sur les performances de ces codes a été abordé.

Pour certains rapports du taux de codage qui satisfont les conditions présentées

dans ce chapitre, les performances des codes turbo sont plutôt médiocres. Une solution a été proposée afin de permettre à ces dernières de se rapprocher assez de la limite de Shannon. Toutefois, pour tous les autres taux de codage, il serait bien d'appliquer la technique de perforation classique comme celle présentée dans le mémoire de Naouefel [11].

Par ailleurs, la nouvelle technique de perforation présentée dans ce chapitre pourrait être encore améliorée, telles que par exemple la recherche du choix idéal des symboles retenus dans le patron modifié. Un exemple de cette recherche est montré dans l'annexe V.

Conclusion et suggestions pour les recherches futures

Ce mémoire a été consacré à l'étude des performances des codes turbo qui sont actuellement parmi les codes les plus performants connus à ce jour.

Dans un premier temps, nous avons concentré notre étude sur des généralités et des rappels de la théorie du codage convolutionnel afin de constituer la base pour tout ce qui suit. Plus particulièrement, nous nous sommes intéressés tout d'abord aux notions fondamentales qui se rattachent aux codes Turbo. Nous avons présenté les codeurs convolutionnels systématiques récursifs ou non et leurs propriétés respectives.

Par la suite, nous avons examiné les différents éléments qui composent la partie codage des codes turbo. La concaténation parallèle des codes convolutionnels systématiques et récursifs, à la base des codes turbo, a été introduite. Après avoir présenté un des éléments très importants des codes turbo, l'entrelaceur avec ses différentes variantes, nous avons étudié la borne union des codes turbo et nous avons introduit un nouvel algorithme qui permet la détermination des distributions de poids de ces codes. Nous avons montré comment ce nouvel algorithme représente mieux la concaténation parallèle du code turbo. Les résultats de simulation ont bien montré aussi qu'il n'y pas une grande différence entre notre algorithme et celui de Benedetto *et al* [6].

Dans le chapitre 3, nous avons présenté ce qui fait la spécificité des codes turbo: le décodage itératif turbo. Nous avons décrit les algorithmes de décodages utilisés par les codes turbo. Par la suite, nous avons étudié l'importance de l'information échangée entre les décodeurs du système turbo et ses effets sur les performances de

ce dernier. Plus la taille des entrelaceurs (selon leurs types) est élevée, plus cette information, appelée information extrinsèque, est décorrélée des autres entrées de prochain décodeur. Ainsi, l'efficacité de correction est meilleure. L'analyse des résultats de simulation dans ce chapitre a montré l'importance du choix des codes convolutionnels systématiques récursifs selon l'intervalle des valeurs de E_b/N_0. Nous avons montré qu'un choix arbitraire des codes élémentaires peut introduire une augmentation de la complexité sans aucune amélioration de la performance des codes turbo en terme de probabilité d'erreur par bit.

Le nouvel entrelaceur à double décalage cyclique que nous avons proposé a la particularité de décaler les lignes, puis les colonnes, de façon alternée vers le haut et vers le bas. Il permet d'obtenir de meilleures performances des codes turbo que celui à simple décalage cyclique. Dans l'annexe III, nous présentons une analyse comparative montrant que les meilleurs résultats sont obtenus avec $D = 2$.

Par ailleurs, nous avons fait une étude comparative entre les différents critères d'arrêt aussi bien dans leurs versions classiques que modifiées. Nous avons trouvé que les meilleurs critères d'arrêt sont: HDAM et HDAMixed, puisqu'ils interrompent le processus de décodage avec moins de délai et de complexité que les autres critères.

Le chapitre 4 a été consacré à la présentation de l'architecture parallèle au niveau du décodage turbo. Une étude détaillée de ce type de système a été faite afin de diminuer le délai du processus de traitement des codes turbo. Nous avons montré que seul pour un taux de codage égal à 1/4, l'architecture parallèle permet d'avoir des gains énormes au niveau du délai de décodage turbo. Dans une autre partie de ce chapitre nous avons présenté les critères d'arrêt adaptés à l'architecture parallèle du décodage turbo. Les résultats de simulation montrent une nouvelle fois que le critère HDA-P qui ressemble à HDAMixed est le meilleur puisque d'une part il est moins complexe et d'autre part il fournit un délai de décodage global faible

par rapport aux autres critères.

Une analyse particulière des codes turbo perforés a été présentée dans le chapitre 5. Elle nous a permis de déterminer comment doit être choisie la matrice de perforation pour certains taux de codage. En effet, nous avons trouvé des relations de dépendance entre le taux de codage des codes turbo, la longueur de contrainte K et la longueur de la réponse impulsionnelle du codeur élémentaire CRS. Si une certaine condition de multiplicité est vérifiée, alors une propriété des patrons de perforation a été identifiée permettant entre autres d'améliorer les performances des codes turbo perforés. Afin de prouver notre point de vue, des résultats de simulation ont été présentés dans les annexes qui suivent. Cependant, pour les autres taux de codage qui ne satisfont pas la condition de multiplicité, l'ancienne méthode de perforation fournit de meilleures performances pour les codes turbo perforés.

Ce travail peut encore être poursuivi, d'une part par une étude approfondie sur la borne union de probabilité afin de comprendre davantage les codes turbo et d'autre part par une analyse permettant la réduction de la complexité de l'algorithme de décodage MAP. Il serait bon également:

- D'étudier les effets combinés de la modulation multiniveaux et la nouvelle méthode de perforation en essayant d'améliorer davantage les performances des codes turbo.

- D'approfondir encore la recherche sur les entrelaceurs adaptés aux codes [49] qui permettent d'augmenter d'une part les distances de Hamming des mots de code et d'autre part la décorrélation entre les séquences de symboles à l'entrée des décodeurs.

- D'adapter les codes turbo à tous les services de multimédia mobile.

- D'améliorer encore la recherche de meilleures positions des symboles retenus dans le nouveau patron de perforation des codes turbo.

Bibliographie

[1] Bahl, L. R., Cocke, J., Jelinek, F., and Raviv, J., "Optimal decoding of linear codes for minimizing symbol error rate", *IEEE Trans. Inf. Theory*, pp. 284-287, Mar. 1974.

[2] Bégin, G., and Haccoun, D., "High-rate punctured convolutional codes: structure properties and construction technique" *IEEE, Trans. commun.* vol.37, No.12, pp. 1381-1385, Dec. 1989.

[3] Bégin, G., "Contributions à l'étude des codes convolutionnels perforés", *Thèse de doctorat*, Ecole Polytechnique de Montréal, Mar. 1990.

[4] Bégin, G., Haccoun, D., and Paquin, C., "Further results on high-rate punctured convolutional codes for Viterbi and sequential decoding" *IEEE, Trans. commun.* vol.38, No.11, pp. 1922-1928, Nov. 1990.

[5] Benedetto, S., and Montorsi, G., "Design of parallel concatenated convolutional codes", *IEEE Trans. commun.*, vol. 42, no 2, pp. 409-428, Mar. 1996.

[6] Benedetto, S., and Montorsi, G., "Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes", *IEEE Trans. Inform. Theory*, vol. 44, no 5, pp. 591-600, May. 1996.

[7] Benedetto, S., Divsalar, D., Montorsi, G., and Pollara, F., "Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding", *JPL TDA Progress Report*, 42-126, pp. 1-26, Aug. 15-1996.

[8] Berrou, C., Glavieux, A., and Thitimajshima P., "Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes", *Proceedings of ICC*, Geneva, Switzerland, pp. 1064-1070, May 1993.

[9] Berrou, C., and Glavieux, A. , "Near Optimum Error Correcting Coding And Decoding: Turbo Codes", *IEEE Trans. Commun.*, vol 44, no. 10, pp. 1261-1271, Oct 1996.

[10] Bhargava, V., Haccoun, D., *Digital Communications By Satellite*, New York: John Wiley, 1981.

[11] Bouzouita, N., "Sur le décodage itératif des codes turbo", *mémoire de maîtrise*, Ecole Polytechnique de Montréal, juin 1997.

[12] Branka, V., and Jinhong, Y., "Turbo Codes: Principles and Applications", *Kluwer Academic Publishers*, 2000.

[13] Divsalar, D., and Pollara, F., "Turbo Codes for Deep-Space Communications", *JPL TDA Progress Report*, 42-120, pp. 29-39, Feb. 1995.

[14] Divsalar, D., and Pollara, F., "Turbo Codes for PCS applications", *in Proc. ICC'95*, Seattle, WA, pp. 59-59, Jun. 1995.

[15] Divsalar, D., Dolinar S., Pollara, F., and McEliece, J., "Transfer function bounds on the performance of turbo codes", *JPL TDA Progress Report*, 42-122, Pasadena, CA, pp. 44-55, Aug. 15,1995.

[16] Divsalar, D., and McEliece, R. J., "The effective free distance of turbo codes", *Electron. lett.*, vol.32, no. 5, pp. 445-446, Feb. 1996.

[17] Dolinar S., and Divsalar, D., "Weight distributions for turbo codes using random and nonrandom permutations", *JPL TDA Progress Report*, 42-122, Pasadena, CA, pp. 56-65, Aug. 15,1995.

[18] Elias, P. "Error-free Coding", *IRE Trans. Inform. Theory*, vol.4, pp. 29-37, 1954.

[19] Erfanian J. A., Pasupathy S., and Gulot G., "Reduced complexity symbol detectors with parallel structures for ISI channels" *IEEE, Trans. commun.*, vol. 42, pp. 1661-1671, 1994.

[20] Fan Mo, Kwatra, S.C., and Junghwan Kim Source, "Analysis of Puncturing Pattern for High Rate Turbo Codes", *IEEE Military Communications. Conference Proceedings*, MILCOM 1999.

[21] Forney, G. D., "Concatenated Codes", *MIT Press*, Cambridge, Mass, 1966

[22] Forney, G. D., "Convolutional codes: algebraic structure", *IEEE Trans. Inform. Theory*, Vol.16, no.6, pp. 720-738, Nov. 1970 et vol.17, no.3, May 1971.

[23] Fossorier, M., Burkert, F., Lin, S., and Hagenauer, J., "On the equivalence between SOVA and Max-Log-MAP decodings", *IEEE Communications Letters*, pp. 137-139, May. 1998.

[24] Haccoun. D., and Bégin, G., "High-rate punctured convolutional codes for Viterbi and sequential decoding" *IEEE, Trans. commun.* vol.37, pp. 1113-1125, Nov. 1989.

[25] Hagenauer, J., and Hoeher, P., "A Viterbi algorithm with soft-decision outputs and its applications", *Globecom*, pp. 1680-1686, Dallas, Texas, USA, Nov. 1989.

[26] Hagenauer, J., Offer, E., and Papke, L., "Iterative decoding of binary block and convolutional codes", *IEEE Trans. Inform. Theory*, vol. 42, no 2, pp. 429-445, Mar. 1996.

[27] Heegard, C., and Wicker, S. B., "Turbo Coding", *Kluwer Academic Publishers*, 1999.

[28] Nyquist, H., "Certain Factors Affecting Telegraph Speed", *Bell Syst. Tech. Jour.*, vol. 3, pp. 324, 1924.

[29] Oberg, M. and Siegel, P.H."The effect of puncturing in turbo encoders" *Proceeding of the Internationnal Symposium on turbo codes and related topics*, pp. 204-207, Brest, France, 1997.

[30] Osseiran A. H., "Sur le décodage des codes turbo" *mémoire de maîtrise*, Ecole Polytechnique de Montréal, Sept. 1999.

[31] Perez, L. C., Seghers, J., and Daniel J. Costello Jr., "A distance spectrum interpretation of turbo codes", *IEEE Trans. Inform. Theory*, vol 42, no. 6, pp. 1698-1709, Nov. 1996.

[32] Pietrobon, S. S., and Barbulescu, S. A., "A Simplification of the Modified Bahl Decoding Algorithm for Systematic Convolutional Codes", *Int. Symp. Inform. Theory and its applications*, pp. 1073-1077, 1994.

[33] Podemski, R., Holubowicz, W., Berrou, C., and Battail, G., "Hamming distance spectra of turbo-codes", *Ann. Telecommun.*, vol. 50, no 9-10, pp. 790-797, Sept.-Oct. 1995.

[34] Proakis, J. G., *Digital Communications*, Third Edition, McGraw-hill, New York, 1995.

[35] Ramsey, J. L., "Realization of optimum interleavers", *IEEE Trans. Inform. Theory*, vol.16, no. 3, pp. 338-345, May. 1970.

[36] Reed, I. S., and Solomon, G., "Polynomial codes over certain finite fields", *SIAM J.*, vol.8, pp. 300-304, 1960.

[37] Robertson, P., "Improving Decoder and Code Structure of Parallel Concatenated Recursive Systematic (Turbo) Codes", *In IEEE Int Conf. Universal Personal Commun.*, San Diego, USA, pp. 183-187, 1994.

[38] Robertson P., Villebrun E., and Hoeher P., "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain" *IEEE International Conference on Communications*, San Diego, USA, pp. 1009-1013, 1995.

[39] Royer, G., "Évaluation des entrelaceurs au sein des codes turbo par simulations", *Mémoire de Maîtrise*, Ecole Polytechnique de Montréal, Nov. 2000

[40] Shannon, C. E., "A Mathematical Theory of Communication", *Bell Syst. Tech. Jour.*, vol. 27, pp. 379-423 (Part I), 623-656 (Part II), Jul-Oct. 1948.

[41] Shannon, C. E., "Probability of error for optimal codes in a gaussian channel", *Bell Syst. Tech. Jour.*, vol. 38, pp. 611-656, 1959.

[42] Svirid, Y. V., "Weight distributions and bounds for turbo codes", *European Trans. Telecommun.*, vol. 6, no 5, pp. 543-556, Sept-Oct 1995.

[43] Shao, R. Y., Lin, S., and Fossorier, M. P. C., "Two Simple stopping Criteria for Turbo Decoding", *IEEE Trans. Commun.*, vol 47, no. 8, pp. 1117-1120, Aug. 1999.

[44] Thitimajshima, P., "Les codes convolutifs récursifs systématiques et leur application à la concaténation parallèle", *Thèse de doctorat*, Université de Bretagne Occidentale, Brest, France, Dec. 1993.

[45] Viterbi, A. J., "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260-269, Apr. 1967.

[46] Vucetic, B., and Yuan, J., *Turbo codes: Principles and applications*, Kluwer Academic Publisher, Boston, 2000.

[47] Wozencraft, J. M., and Jacobs, I. M., *Principles of Communication Engineering*, Wiley, New York, 1965.

[48] Yasuda, Y., Kashiki, K., and Hirata, Y., "High-Rate Punctured Convolutional Codes for Soft Decision Viterbi Decoding", *IEEE Trans. Commun.*, vol. COM-32, pp. 315-319, 1984.

[49] Yuan, J., Vucetic, B., and Feng, W., "Combined turbo codes and interleaver design ", *IEEE Trans. Commun.*, vol. 47, no 4, pp. 484-487, Apr 1999.

[50] Zerong, Y., "Analyse et performance des codes convolutionnels récursifs systématiques ", *Mémoire de Maîtrise*, Ecole Polytechnique de Montréal, 1998.

Annexe I

Représentation matricielle des entrelaceurs

La représentation matricielle des entrelaceurs est couramment utilisée pour observer et analyser l'effort de permutation et la séparation des symboles adjacents. Il s'agit d'une matrice NxN (N est la longueur du bloc à entrelacer c'est à dire la taille de l'entrelaceur) contenant un seul 1 par ligne et colonne, et les autres éléments sont à zéro. En d'autres termes, cela veut dire que si un 1 apparaît dans la $i^{ème}$ ligne et la $j^{ème}$ colonne, alors le $i^{ème}$ symbole de la séquence d'entrée va être le $j^{ème}$ symbole dans la séquence de sortie.

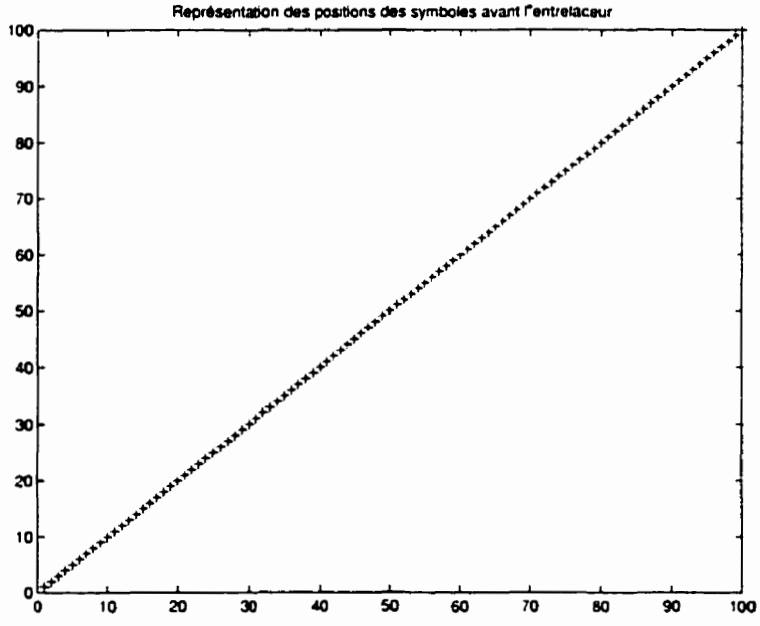


Figure I.1: Graphe d'une séquence de symboles de taille 100 non entrelacée

La matrice représentative d'un entrelaceur peut être illustrée par un graphe où chaque position d'un 1 sera présentée par une croix. De cette façon, il sera possible

de visualiser la structure exacte d'un entrelaceur. Ainsi, pour des symboles consécutifs dans une séquence à l'entrée, plus leurs croix représentatives sont éloignées plus l'entrelaceur est bon.

Sans entrelacement la séquence à l'entrée peut être représentée par une matrice identité c'est-à-dire un graphe où les croix suivent la même diagonale comme l'indique la figure I.1.

Dans la suite nous allons dresser les graphes de tous les entrelaceurs mentionnés dans le chapitre 2 de taille égale à $N = 100$.

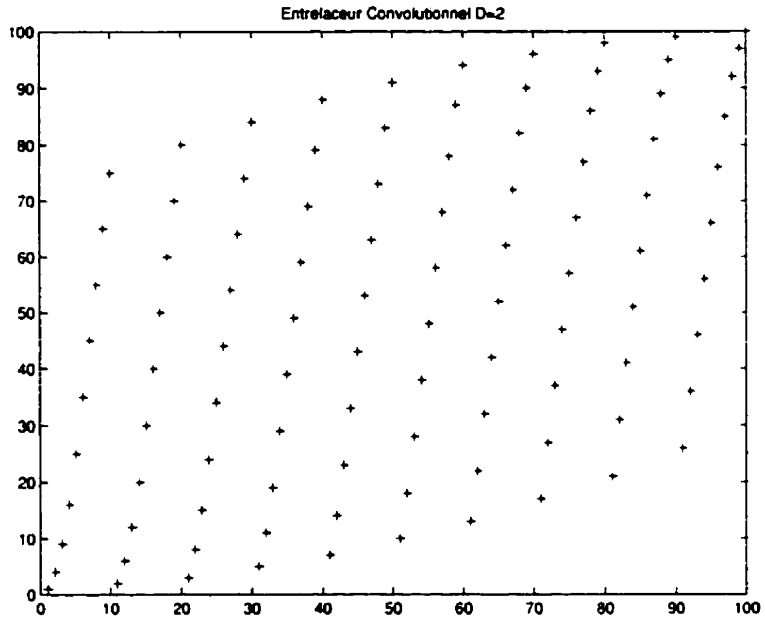


Figure I.2: Représentation matricielle de l'entrelaceur convolutionnel

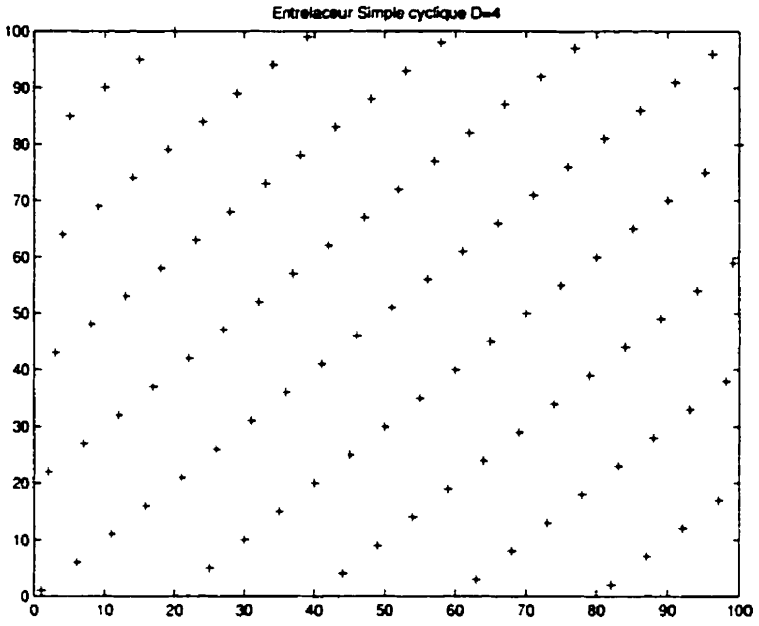


Figure I.3: Représentation matricielle de l'entrelaceur à décalage simple cyclique

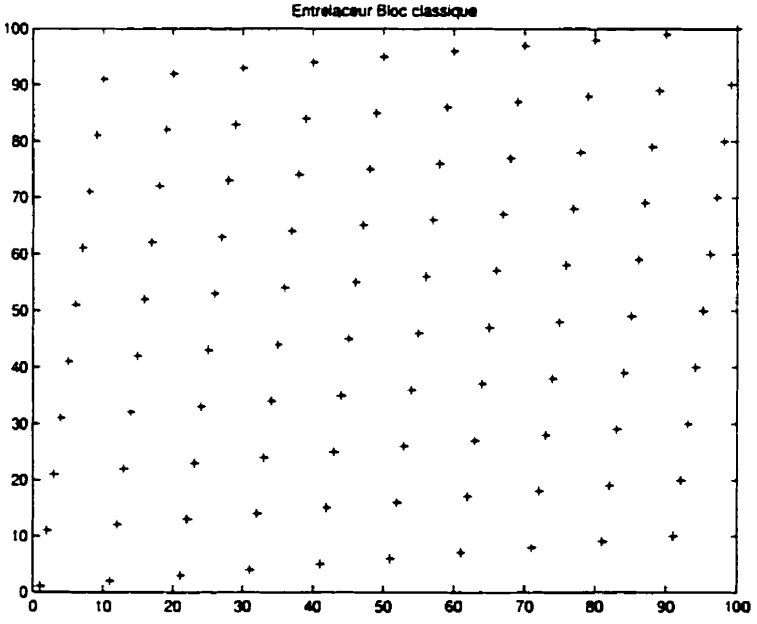


Figure I.4: Représentation matricielle de l'entrelaceur Bloc

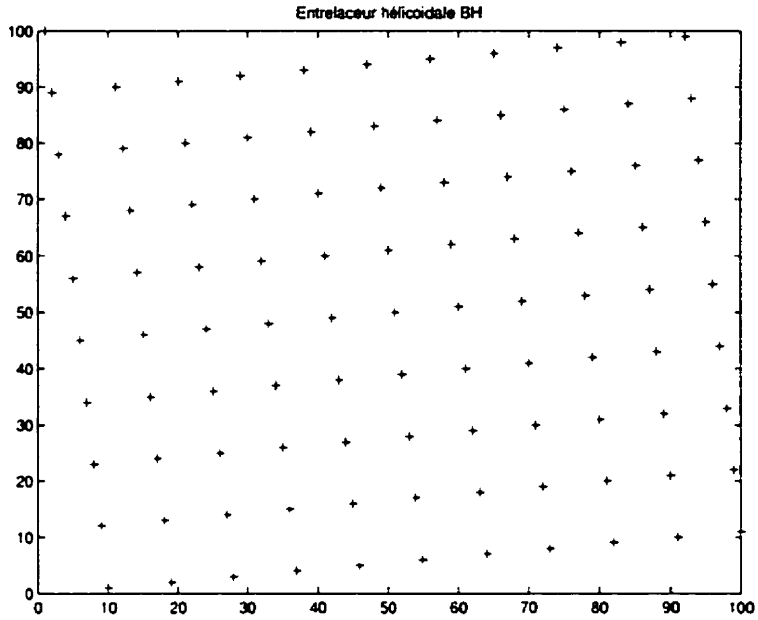


Figure I.5: Représentation matricielle de l'entrelaceur hélicoïdal BH

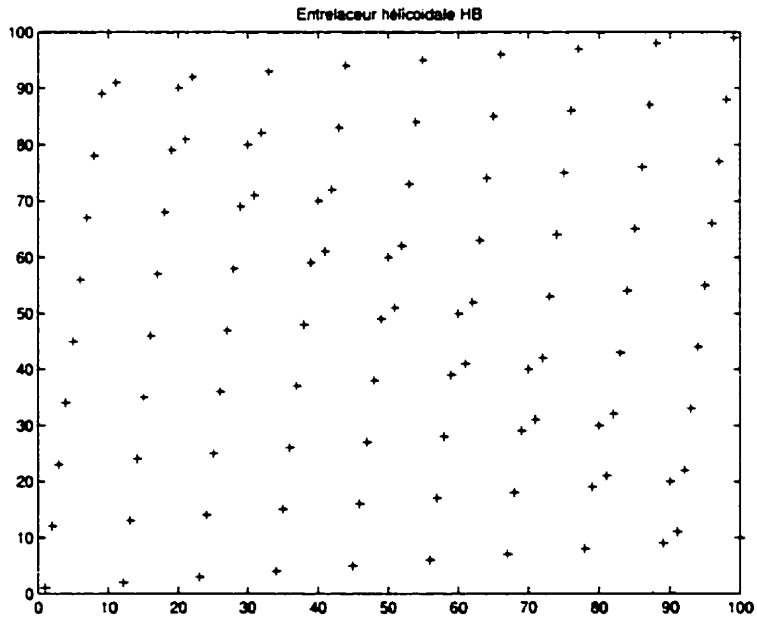


Figure I.6: Représentation matricielle de l'entrelaceur hélicoïdal HB

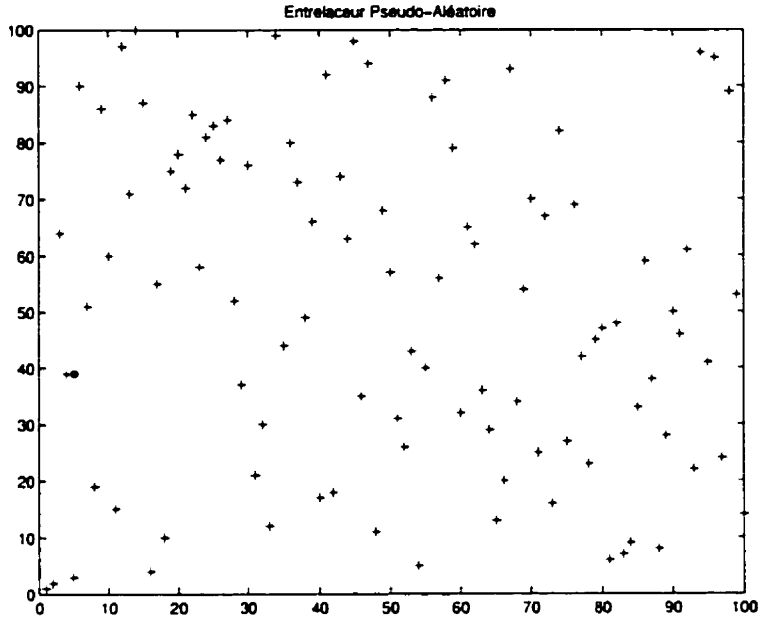


Figure I.7: Représentation matricielle de l'entrelaceur aléatoire

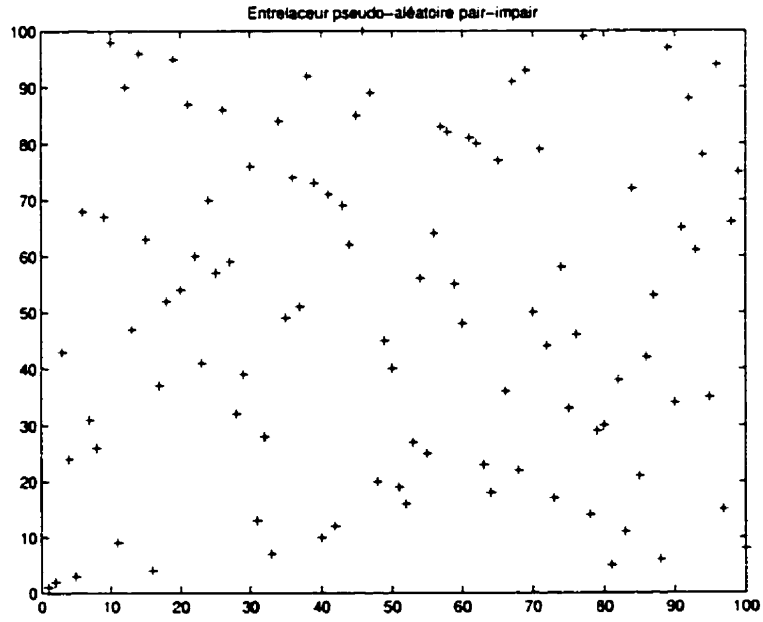


Figure I.8: Représentation matricielle de l'entrelaceur pseudo-aléatoire pair-impair

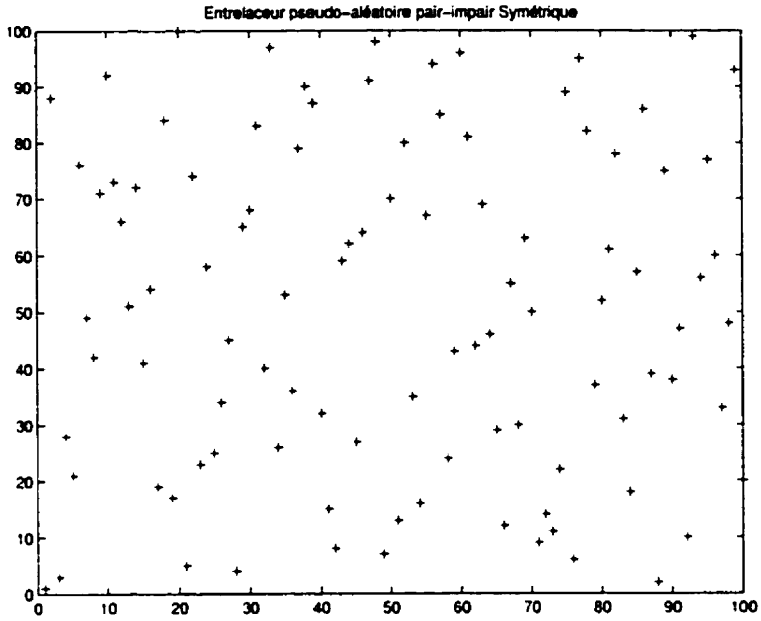


Figure I.9: Représentation matricielle de l'entrelaceur pseudo-aléatoire pair-impair symétrique

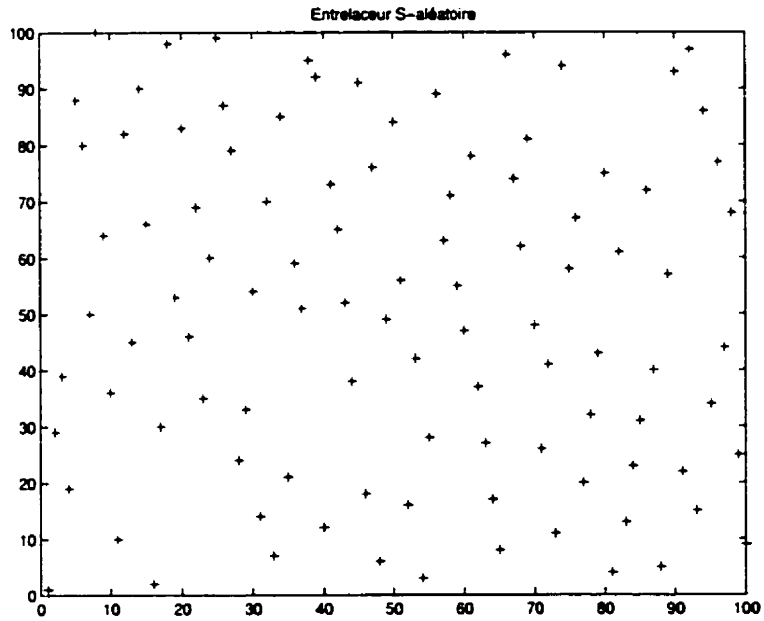


Figure I.10: Représentation matricielle de l'entrelaceur S-aléatoire

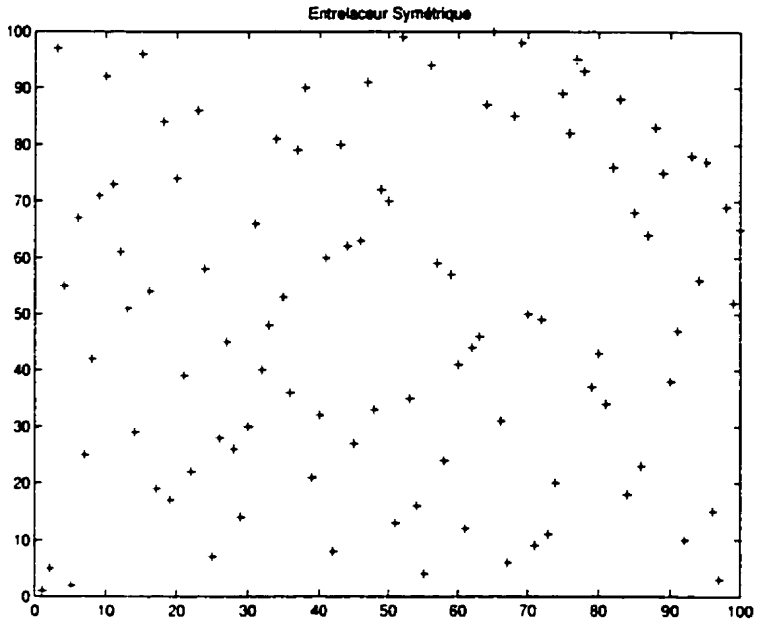


Figure I.11: Représentation matricielle de l'entrelaceur symétrique

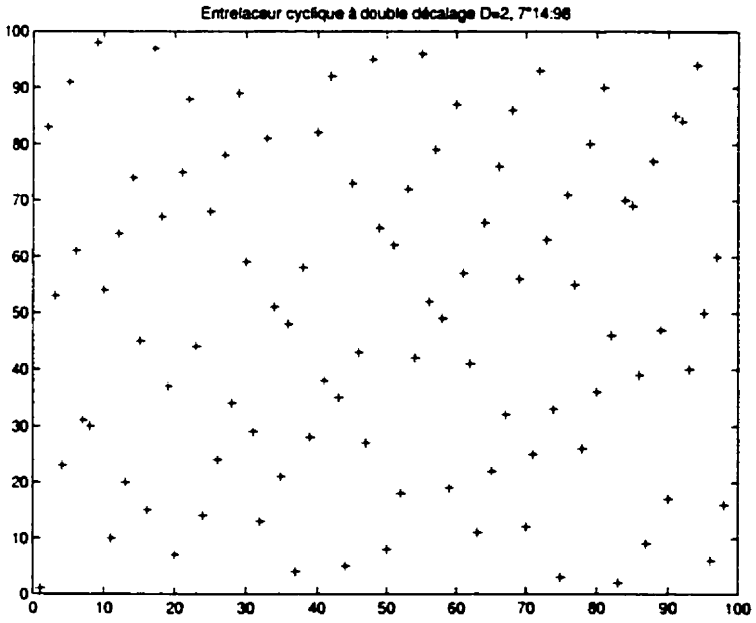


Figure I.12: Représentation matricielle de l'entrelaceur à décalage double cyclique

Annexe II

Algorithme MAP

II.1 Description de l'algorithme:

On fera constamment usage au cours de cette annexe de résultats théoriques ou pratiques dont la source peut être retracée dans plusieurs ouvrages sur les algorithmes BCJR et MAP, [1], [8], [37].

L'algorithme de MAP est un algorithme optimal. Nous allons présenter plus particulièrement, l'algorithme MAP qui calcul la probabilité à posteriori du symbole d'information encodé par un codeur convolutionnel CRS de taux de codage $R = 1/2$ et transmis d'une part dans un canal AWGN et d'autre part dans un canal de Rayleigh.

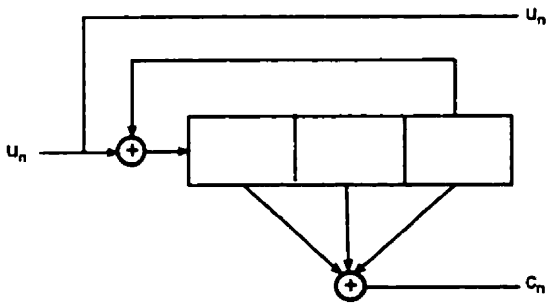


Figure II.1: Exemple d'un codeur convolutionnel récursif systématique $R = 1/2, K = 3, G = [1, 7/5]$

On désigne par $\mathbf{U}_1^N = (u_1, u_2, \ldots, u_n, \ldots, u_N)$ et $\mathbf{C}_1^N = (c_1, c_2, \ldots, c_n, \ldots, c_N)$ les séquences binaires respectivement systématique et parité du codeur illustré par la figure II.1.

Soient $\mathbf{X}_1^N = (x_1, x_2, \ldots, x_n, \ldots, x_N)$ et $\mathbf{Y}_1^N = (y_1, y_2, \ldots, y_n, \ldots, y_N)$ les séquences reçues associées respectivement à $\mathbf{U}_1^N$ et $\mathbf{C}_1^N$. Chaque éléments des vecteurs $\mathbf{X}_1^N$ et

$\mathbf{Y}_1^N$ peut être modélisé par :

$$x_n = (2u_n - 1) + w_{x_n} \quad \text{(II.1)}$$

$$y_n = (2c_n - 1) + w_{y_n} \quad \text{(II.2)}$$

où w_{x_n} et w_{y_n} sont des variables aléatoires Gaussiennes et indépendantes ($N(0, \sigma^2)$), de moyenne nulle et de variance σ^2 .

A part les séquences $\mathbf{X}_1^N$ et $\mathbf{Y}_1^N$, le décodeur a l'accès à une autre ressources qui est la séquence d'information à priori $\tilde{\Lambda}_1^N$. Chaque élément de cette séquence peut être représenté en domaine logarithmique par :

$$\tilde{\Lambda}_n = \ln \frac{Pr(u_n = 1)}{Pr(u_n = 0)} \quad \text{(II.3)}$$

Chaque décodeur dispose de trois séquences qu'on peut les noter par un simple vecteur $\mathbf{R}_1^N$ tel que:

$$\mathbf{R}_1^N = (\mathbf{X}_1^N, \mathbf{Y}_1^N, \tilde{\Lambda}_1^N) = (\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_n, \ldots, \mathbf{R}_N) \text{ avec } \mathbf{R}_n = (x_n, y_n, \tilde{\Lambda}_n) \quad \text{(II.4)}$$

L'objectif principal d'un algorithme MAP est le calcule des probabilités à posteriori du symbole $u_n = 0$ où 1 conditionnellement sur $\mathbf{R}_1^N$, c'est à dire $Pr(u_n = 1|\mathbf{R}_1^N)$ et $Pr(u_n = 0|\mathbf{R}_1^N)$. Ces probabilités sont en fonction des probabilités de transition des états. On pose $S_{n-1} \in \{0, 1, \ldots, 2^M - 1\}$ l'état du codeur après $n-1$ symboles binaires. Par la suite, l'introduction d'un symbole binaire u_n entraîne la transition de l'état du codeur de S_{n-1} vers S_n. Dans notre exemple, nous avons 4 états possibles comme l'indique la figure II.2 qui montre les différentes transitions entre l'instant $n-1$ et n.

La probabilité à posteriori peut s'exprimer par:

$$Pr(u_n = i|\mathbf{R}_1^N) = \sum_{s'=0}^{2^M-1} \sum_{s=0}^{2^M-1} Pr(u_n = i, S_{n-1} = s', S_n = s|\mathbf{R}_1^N), \qquad i = 0, 1. \quad \text{(II.5)}$$

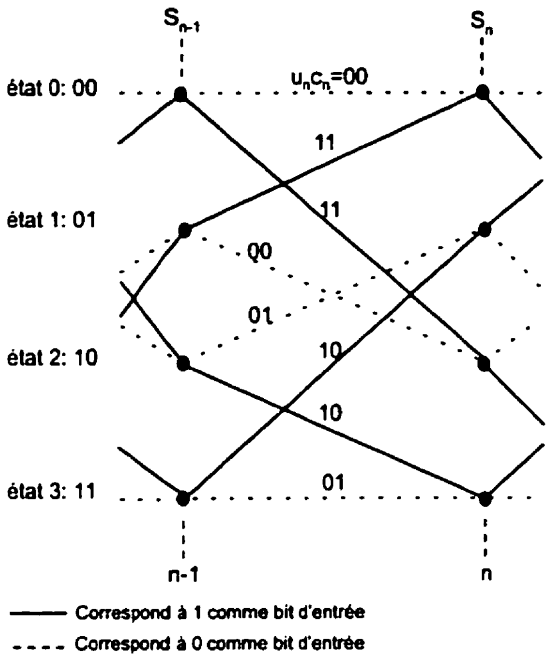


Figure II.2: Diagramme en treillis de la figure II.1

Pour mieux faciliter les calculs, il est bien d'introduire la probabilité conjointe σ_n^i définie par:

$$\sigma_n^i(s', s) = Pr(u_n = i, S_{n-1} = s', S_n = s, \mathbf{R}_1^N) \tag{II.6}$$

(II.5) devient alors:

$$Pr(u_n = i|\mathbf{R}_1^N) = \sum_{s'=0}^{2^M-1} \sum_{s=0}^{2^M-1} \frac{\sigma_n^i(s', s)}{Pr(\mathbf{R}_1^N)}, \qquad i = 0, 1. \tag{II.7}$$

On peut encore simplifier $\sigma_n^i(s', s)$ par:

$$\begin{aligned}
\sigma_n^i(s', s) &= Pr(u_n = i, S_{n-1} = s', S_n = s, \mathbf{R}_1^n, \mathbf{R}_{n+1}^N) \\
&= Pr(u_n = i, S_{n-1} = s', S_n = s, \mathbf{R}_1^n) \ . \\
&\quad Pr(\mathbf{R}_{n+1}^N | u_n = i, S_{n-1} = s', S_n = s, \mathbf{R}_1^n) \\
&= Pr(u_n = i, S_{n-1} = s', S_n = s, \mathbf{R}_1^{n-1}, \mathbf{R}_n) \ .
\end{aligned}$$

$$
\begin{aligned}
& Pr(\mathbf{R}_{n+1}^{N}|u_n = i, S_{n-1} = s', S_n = s, \mathbf{R}_1^n) \\
= \; & Pr(S_{n-1} = s', \mathbf{R}_1^{n-1}) Pr(u_n = i, S_n = s, \mathbf{R}_n|S_{n-1} = s', \mathbf{R}_1^{n-1}) \cdot \\
& Pr(\mathbf{R}_{n+1}^{N}|u_n = i, S_{n-1} = s', S_n = s, \mathbf{R}_1^n) \qquad \text{(II.8)}
\end{aligned}
$$

Si on connaît l'état S_{n-1} alors le bit u_n, l'état S_n et le vecteur $\mathbf{R}_n$ seront indépendants de $\mathbf{R}_1^{n-1}$. Par conséquent,

$$Pr(u_n = i, S_n = s, \mathbf{R}_n|S_{n-1} = s', \mathbf{R}_1^{n-1}) = Pr(u_n = i, S_n = s, \mathbf{R}_n|S_{n-1} = s')$$

Par analogie, on trouve aussi

$$Pr(\mathbf{R}_{n+1}^{N}|u_n = i, S_{n-1} = s', S_n = s, \mathbf{R}_1^n) = Pr(\mathbf{R}_{n+1}^{N}|S_n = s)$$

Ainsi, $\sigma_n^i(s', s)$ peut s'écrire:

$$
\begin{aligned}
\sigma_n^i(s', s) &= Pr(S_{n-1} = s', \mathbf{R}_1^{n-1}) Pr(u_n = i, S_n = s, \mathbf{R}_n|S_{n-1} = s') \\
&\quad Pr(\mathbf{R}_{n+1}^{N}|S_n = s) \\
&= \tilde{\alpha}_{n-1}(s') \tilde{\gamma}_n^i(\mathbf{R}_n, s', s) \tilde{\beta}_n(s) \qquad \text{(II.9)}
\end{aligned}
$$

où les quantités $\tilde{\alpha}_{n-1}(s')$, $\tilde{\gamma}_n^i(\mathbf{R}_n, s', s)$ et $\tilde{\beta}_n(s)$ sont définies par:

$$
\begin{aligned}
\tilde{\alpha}_{n-1}(s') &= Pr(S_{n-1} = s', \mathbf{R}_1^{n-1}) \qquad \text{(II.10)} \\
\tilde{\gamma}_n^i(\mathbf{R}_n, s', s) &= Pr(u_n = i, S_n = s, \mathbf{R}_n|S_{n-1} = s') \qquad \text{(II.11)} \\
\tilde{\beta}_n(s) &= Pr(\mathbf{R}_{n+1}^{N}|S_n = s) \qquad \text{(II.12)}
\end{aligned}
$$

Les quantités $\tilde{\alpha}$, $\tilde{\gamma}$ et $\tilde{\beta}$ sont des probabilités appelées respectivement métrique d'état en avant, métrique de branche et métrique d'état en arrière. Elles sont particulièrement reliées à la probabilité de transition des états du codeur et la probabilité de transition du canal. Dans la suite on va détailler les expressions de ces différentes quantités afin de simplifier le calcul du rapport de vraisemblance

associé à chaque symbole d'information u_n. On obtient:

$$\begin{aligned}
\tilde{\alpha}_n(s) &= Pr(S_n = s, \mathbf{R}_1^n) \\
&= \sum_{s'=0}^{2^M-1} \sum_{i=0}^{1} Pr(u_n = i, S_{n-1} = s', S_n = s, \mathbf{R}_1^n) \\
&= \sum_{s'=0}^{2^M-1} \sum_{i=0}^{1} Pr(u_n = i, S_{n-1} = s', S_n = s, \mathbf{R}_1^{n-1}, \mathbf{R}_n) \\
&= \sum_{s'=0}^{2^M-1} \sum_{i=0}^{1} Pr(S_{n-1} = s', \mathbf{R}_1^{n-1}) Pr(u_n = i, S_n = s, \mathbf{R}_n | S_{n-1} = s', \mathbf{R}_1^{n-1}) \\
&= \sum_{s'=0}^{2^M-1} \sum_{i=0}^{1} Pr(S_{n-1} = s', \mathbf{R}_1^{n-1}) Pr(u_n = i, S_n = s, \mathbf{R}_n | S_{n-1} = s') \\
&= \sum_{s'=0}^{2^M-1} \sum_{i=0}^{1} \tilde{\alpha}_{n-1}(s') \tilde{\gamma}_n^i(\mathbf{R}_n, s', s) \qquad \text{(II.13)}
\end{aligned}$$

et

$$\begin{aligned}
\tilde{\beta}_n(s') &= Pr(\mathbf{R}_{n+1}^N | S_n = s') \\
&= \sum_{s=0}^{2^M-1} \sum_{i=0}^{1} Pr(u_{n+1} = i, S_{n+1} = s, \mathbf{R}_{n+1}^N | S_n = s') \\
&= \sum_{s=0}^{2^M-1} \sum_{i=0}^{1} Pr(u_{n+1} = i, S_{n+1} = s, \mathbf{R}_{n+1}, \mathbf{R}_{n+2}^N | S_n = s') \\
&= \sum_{s=0}^{2^M-1} \sum_{i=0}^{1} Pr(\mathbf{R}_{n+2}^N | u_{n+1} = i, S_{n+1} = s, \mathbf{R}_{n+1}, S_n = s') \cdot \\
&\quad \frac{Pr(u_{n+1} = i, S_{n+1} = s, \mathbf{R}_{n+1}, S_n = s')}{Pr(S_n = s')} \\
&= \sum_{s=0}^{2^M-1} \sum_{i=0}^{1} Pr(\mathbf{R}_{n+2}^N | S_{n+1} = s) \cdot \\
&\quad Pr(u_{n+1} = i, S_{n+1} = s, \mathbf{R}_{n+1} | S_n = s') \\
&= \sum_{s=0}^{2^M-1} \sum_{i=0}^{1} \tilde{\beta}_{n+1}(s) \tilde{\gamma}_{n+1}^i(\mathbf{R}_{n+1}, s', s) \qquad \text{(II.14)}
\end{aligned}$$

Il reste maintenant à simplifier d'avantage la fonction de probabilité de $\tilde{\gamma}_n^i(\mathbf{R}_{n+1}, s', s)$. A partir de (II.11), on peut écrire:

$$\begin{aligned}
\tilde{\gamma}_n^i(\mathbf{R}_n, s', s) &= Pr(u_n = i, S_n = s, \mathbf{R}_n | S_{n-1} = s') \\
&= Pr(\mathbf{R}_n | u_n = i, S_n = s, S_{n-1} = s') \ . \\
&\quad Pr(S_n = s | u_n = i, S_{n-1} = s') Pr(u_n = i) \qquad \text{(II.15)}
\end{aligned}$$

Changeant $\mathbf{R}_n$ par $(x_n, y_n, \tilde{\Lambda}_n)$ et supposant que les entrées de décodeur sont indépendantes entre elles conditionnellement sur $u_n = i$, $S_n = s$ et $S_{n-1} = s'$, alors:

$$\begin{aligned}
\tilde{\gamma}_n^i(\mathbf{R}_n, s', s) &= Pr(x_n, y_n, \tilde{\Lambda}_n | u_n = i, S_n = s, S_{n-1} = s') \ . \\
&\quad Pr(S_n = s | u_n = i, S_{n-1} = s') Pr(u_n = i) \\
&= Pr(x_n | u_n = i, S_n = s, S_{n-1} = s') \ . \\
&\quad Pr(y_n | u_n = i, S_n = s, S_{n-1} = s') \ . \\
&\quad Pr(\tilde{\Lambda}_n | u_n = i, S_n = s, S_{n-1} = s') \ . \\
&\quad Pr(S_n = s | u_n = i, S_{n-1} = s') Pr(u_n = i) \\
&= Pr(x_n | u_n = i) Pr(y_n | u_n = i, S_{n-1} = s') \ . \\
&\quad Pr(\tilde{\Lambda}_n | u_n = i) Pr(u_n = i) \ . \\
&\quad Pr(S_n = s | u_n = i, S_{n-1} = s') \\
&= Pr(x_n | u_n = i) Pr(y_n | u_n = i, S_{n-1} = s') \ . \\
&\quad Pr(u_n = i | \tilde{\Lambda}_n) Pr(\tilde{\Lambda}_n) \ . \\
&\quad Pr(S_n = s | u_n = i, S_{n-1} = s') \qquad \text{(II.16)}
\end{aligned}$$

II.2 Cas d'un canal AWGN:

x_n et y_n sont considérées comme deux variables aléatoires gaussiennes de moyennes respectives $(2u_n - 1)$ et $(2c_n - 1)$ et de variance σ^2. Ainsi,

$$Pr(x_n|u_n = i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_n-(2i-1))^2}{2\sigma^2}} \quad \text{(II.17)}$$

$$Pr(y_n|u_n = i, S_{n-1} = s') = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_n-(2c_n-1))^2}{2\sigma^2}} \quad \text{(II.18)}$$

En outre, $Pr(u_n = i|\tilde{\Lambda}_n)$ est l'information à priori du symbole u_n. En utilisant (II.3) et l'égalité $Pr(u_n = 1) = 1 - Pr(u_n = 0)$, on déduit:

$$Pr(u_n = i|\tilde{\Lambda}_n) = \frac{e^{i\tilde{\Lambda}_n}}{1 + e^{\tilde{\Lambda}_n}} \quad \text{Autrement dit,} \quad \text{(II.19)}$$

$$Pr(u_n = 1|\tilde{\Lambda}_n) = \frac{e^{\tilde{\Lambda}_n}}{1 + e^{\tilde{\Lambda}_n}} \quad \text{et} \quad \text{(II.20)}$$

$$Pr(u_n = 0|\tilde{\Lambda}_n) = \frac{1}{1 + e^{\tilde{\Lambda}_n}} \quad \text{(II.21)}$$

Revenant à (II.16), on a d'une part la probabilité $Pr(\tilde{\Lambda}_n)$ qui est une variable constante par rapport à la valeur de u_n et d'un autre part la probabilité $Pr(S_n = s|u_n = i, S_{n-1} = s')$ qui est égale à 0 ou à 1. Elle dépend s'il y a ou non une transition dans le treillis entre l'état s' et l'état s. Par conséquent, le rapport de vraisemblance peut être écrit:

$$\begin{aligned}
\Lambda_n &= \ln \frac{Pr(u_n = 1|\mathbf{R}_1^N)}{Pr(u_n = 0|\mathbf{R}_1^N)} \\
&= \ln \frac{\sum_{s'} \sum_s \tilde{\alpha}_{n-1}(s')\tilde{\gamma}_n^1(\mathbf{R}_n, s', s)\tilde{\beta}_n(s)}{\sum_{s'} \sum_s \tilde{\alpha}_{n-1}(s')\tilde{\gamma}_n^0(\mathbf{R}_n, s', s)\tilde{\beta}_n(s)} \\
&= \ln \frac{\sum_{s'} \sum_s \tilde{\alpha}_{n-1}(s')Pr(x_n|u_n = 1)Pr(y_n|u_n = 1, S_{n-1} = s')}{\sum_{s'} \sum_s \tilde{\alpha}_{n-1}(s')Pr(x_n|u_n = 0)Pr(y_n|u_n = 0, S_{n-1} = s')} \\
&\quad \frac{Pr(u_n = 1|\tilde{\Lambda}_n)Pr(\tilde{\Lambda}_n)Pr(S_n = s|u_n = 1, S_{n-1} = s')\tilde{\beta}_n(s)}{Pr(u_n = 0|\tilde{\Lambda}_n)Pr(\tilde{\Lambda}_n)Pr(S_n = s|u_n = 0, S_{n-1} = s')\tilde{\beta}_n(s)} \\
&= \ln \frac{Pr(x_n|u_n = 1)Pr(u_n = 1|\tilde{\Lambda}_n)\sum_{s'} \sum_s \tilde{\alpha}_{n-1}(s')}{Pr(x_n|u_n = 0)Pr(u_n = 0|\tilde{\Lambda}_n)\sum_{s'} \sum_s \tilde{\alpha}_{n-1}(s')} \\
&\quad \frac{Pr(y_n|u_n = 1, S_{n-1} = s')Pr(S_n = s|u_n = 1, S_{n-1} = s')\tilde{\beta}_n(s)}{Pr(y_n|u_n = 0, S_{n-1} = s')Pr(S_n = s|u_n = 0, S_{n-1} = s')\tilde{\beta}_n(s)} \quad \text{(II.22)}
\end{aligned}$$

Par analogie avec (3.6), nous pouvons écrire:

$$\begin{aligned}
\Lambda_n &= \ln \frac{Pr(x_n|u_n=1)Pr(u_n=1|\tilde{\Lambda}_n)\sum_{s'}\sum_s \tilde{\alpha}_{n-1}(s')}{Pr(x_n|u_n=0)Pr(u_n=0|\tilde{\Lambda}_n)\sum_{s'}\sum_s \tilde{\alpha}_{n-1}(s')} \\
&\quad \frac{Pr(y_n|u_n=1,S_{n-1}=s')Pr(S_n=s|u_n=1,S_{n-1}=s')\tilde{\beta}_n(s)}{Pr(y_n|u_n=0,S_{n-1}=s')Pr(S_n=s|u_n=0,S_{n-1}=s')\tilde{\beta}_n(s)} \\
&= \ln \frac{Pr(x_n|u_n=1)}{Pr(x_n|u_n=0)} + \ln \frac{Pr(u_n=1|\tilde{\Lambda}_n)}{Pr(u_n=0|\tilde{\Lambda}_n)} + \\
&\quad \ln \frac{Pr(\mathbf{X}_1^{n-1},\mathbf{X}_{n+1}^N,\mathbf{Y}_1^N,\tilde{\mathbf{\Lambda}}_1^{n-1},\tilde{\mathbf{\Lambda}}_{n+1}^N|u_n=1)}{Pr(\mathbf{X}_1^{n-1},\mathbf{X}_{n+1}^N,\mathbf{Y}_1^N,\tilde{\mathbf{\Lambda}}_1^{n-1},\tilde{\mathbf{\Lambda}}_{n+1}^N|u_n=0)} \\
&= \frac{2}{\sigma^2}x_n + \tilde{\Lambda}_n + Le_n
\end{aligned} \tag{II.23}$$

où Le_n est l'information extrinsèque qui peut s'exprimer par:

$$\begin{aligned}
Le_n &= \Lambda_n - \frac{2}{\sigma^2}x_n - \tilde{\Lambda}_n \\
&= \ln \frac{\sum_{s'}\sum_s \tilde{\alpha}_{n-1}(s')\tilde{\gamma}_n^1(\mathbf{R}_n,s',s)\tilde{\beta}_n(s)}{\sum_{s'}\sum_s \tilde{\alpha}_{n-1}(s')\tilde{\gamma}_n^0(\mathbf{R}_n,s',s)\tilde{\beta}_n(s)} - \frac{2}{\sigma^2}x_n - \tilde{\Lambda}_n
\end{aligned} \tag{II.24}$$

II.3 Cas d'un canal de Rayleigh:

Dans ce cas, x_n et y_n sont deux variables aléatoires qui suivent une distribution de Rayleigh de moyennes respectives $m_\alpha \times (2u_n - 1)$ et $m_\alpha \times (2c_n - 1)$ avec $m_\alpha = 0.8862$. Par conséquent, on a:

$$Pr(x_n|u_n=i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{m_\alpha(x_n-(2i-1))^2}{2\sigma^2}} \tag{II.25}$$

$$Pr(y_n|u_n=i,S_{n-1}=s') = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{m_\alpha(y_n-(2c_n-1))^2}{2\sigma^2}} \tag{II.26}$$

Ainsi, (II.23) s'exprime après ce changement par:

$$\Lambda_n = m_\alpha \frac{2}{\sigma^2}x_n + \tilde{\Lambda}_n + Le_n \tag{II.27}$$

II.4 Implémentation:

II.4.1 Dans un canal AWGN

En combinant (II.17), (II.18), (II.19) et (II.16), on peut encore simplifier $\tilde{\gamma}_n^i(\mathbf{R}_n, s', s)$ par:

$$\begin{aligned}\tilde{\gamma}_n^i(\mathbf{R}_n, s', s) &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_n-(2i-1))^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_n-(2c_n-1))^2}{2\sigma^2}} \frac{e^{i\tilde{\Lambda}_n}}{1+e^{\tilde{\Lambda}_n}} \\ &\quad Pr(\tilde{\Lambda}_n) Pr(S_n = s | u_n = i, S_{n-1} = s') \\ &= \frac{Pr(\tilde{\Lambda}_n)}{2\pi\sigma(1+e^{\tilde{\Lambda}_n})} e^{-\frac{x_n^2+(2i-1)^2+y_n^2+(2c_n-1)^2}{2\sigma^2}} \\ &\quad e^{\frac{x_n(2i-1)+y_n(2c_n-1)}{\sigma^2}+i\tilde{\Lambda}_n} Pr(S_n = s | u_n = i, S_{n-1} = s') \quad \text{(II.28)}\end{aligned}$$

Or $(2i-1)^2 = 1$ et $(2c_n - 1)^2 = 1$. Posant aussi :

$$\begin{aligned}C_{\gamma_n} &= \frac{Pr(\tilde{\Lambda}_n)}{2\pi\sigma(1+e^{\tilde{\Lambda}_n})} e^{-\frac{x_n^2+y_n^2+2(x_n+c_n+1)}{2\sigma^2}} \qquad \text{et} \\ \gamma_n^i(\mathbf{R}_n, s', s) &= e^{\frac{2(x_n i+y_n c_n)}{\sigma^2}+i\tilde{\Lambda}_n} Pr(S_n = s | u_n = i, S_{n-1} = s')\end{aligned}$$

On obtient alors:

$$\tilde{\gamma}_n^i(\mathbf{R}_n, s', s) = C_{\gamma_n} \gamma_n^i(\mathbf{R}_n, s', s) \quad \text{(II.29)}$$

Puisque C_{γ_n} est indépendante de s. de s' et de $u_n = i$, on peut réécrire les quantités suivantes: $\tilde{\alpha}_n(s)$ et $\tilde{\beta}_n(s')$ par :

$$\begin{aligned}\tilde{\alpha}_n(s) &= \sum_{s'=0}^{2^M-1} \sum_{i=0}^{1} \tilde{\alpha}_{n-1}(s') \tilde{\gamma}_n^i(\mathbf{R}_n, s', s) \\ &= C_{\gamma_n} \sum_{s'=0}^{2^M-1} \sum_{i=0}^{1} \tilde{\alpha}_{n-1}(s') \gamma_n^i(\mathbf{R}_n, s', s) \quad \text{(II.30)} \\ \tilde{\beta}_n(s') &= \sum_{s=0}^{2^M-1} \sum_{i=0}^{1} \tilde{\beta}_{n+1}(s) \tilde{\gamma}_{n+1}^i(\mathbf{R}_{n+1}, s', s)\end{aligned}$$

$$= C_{\gamma_n} \sum_{s=0}^{2^M-1} \sum_{i=0}^{1} \tilde{\beta}_{n+1}(s)\gamma_{n+1}^{i}(\mathbf{R}_{n+1}, s', s) \qquad \text{(II.31)}$$

Dans le rapport de vraisemblance C_{γ_n} sera simplifiée, donc on pourra redéfinir respectivement $\tilde{\alpha}_n(s)$ et $\tilde{\beta}_n(s')$ par $\alpha_n(s)$ et $\beta_n(s')$:

$$\alpha_n(s) = \sum_{s'=0}^{2^M-1} \sum_{i=0}^{1} \alpha_{n-1}(s')\gamma_n^{i}(\mathbf{R}_n, s', s) \qquad \text{(II.32)}$$

$$\beta_n(s') = \sum_{s=0}^{2^M-1} \sum_{i=0}^{1} \beta_{n+1}(s)\gamma_{n+1}^{i}(\mathbf{R}_{n+1}, s', s) \qquad \text{(II.33)}$$

Et finalement, Λ_n et l'information extrinsèque peuvent s'exprimer par:

$$\Lambda_n = \ln \frac{\sum_{s'} \sum_{s} \alpha_{n-1}(s')\gamma_n^{1}(\mathbf{R}_n, s', s)\beta_n(s)}{\sum_{s'} \sum_{s} \alpha_{n-1}(s')\gamma_n^{0}(\mathbf{R}_n, s', s)\beta_n(s)} \qquad \text{(II.34)}$$

$$Le_n = \ln \frac{\sum_{s'} \sum_{s} \alpha_{n-1}(s')\gamma_n^{1}(\mathbf{R}_n, s', s)\beta_n(s)}{\sum_{s'} \sum_{s} \alpha_{n-1}(s')\gamma_n^{0}(\mathbf{R}_n, s', s)\beta_n(s)} - \frac{2}{\sigma^2} x_n - \tilde{\Lambda}_n \qquad \text{(II.35)}$$

II.4.2 Dans un canal de Rayleigh

Avec (II.25), (II.26), (II.19) et (II.16), $\tilde{\gamma}_n^{i}(\mathbf{R}_n, s', s)$ s'écrit dans un canal de Rayleigh par:

$$\begin{aligned}
\tilde{\gamma}_n^{i}(\mathbf{R}_n, s', s) &= \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{m_a(x_n-(2i-1))^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{m_a(y_n-(2c_n-1))^2}{2\sigma^2}} \frac{e^{i\tilde{\Lambda}_n}}{1+e^{\tilde{\Lambda}_n}} \\
&\quad Pr(\tilde{\Lambda}_n) Pr(S_n = s | u_n = i, S_{n-1} = s') \\
&= \frac{Pr(\tilde{\Lambda}_n)}{2\pi\sigma(1+e^{\tilde{\Lambda}_n})} e^{-m_a \frac{x_n^2+(2i-1)^2+y_n^2+(2c_n-1)^2}{2\sigma^2}} \\
&\quad e^{m_a \frac{x_n(2i-1)+y_n(2c_n-1)}{\sigma^2}+i\tilde{\Lambda}_n} Pr(S_n = s | u_n = i, S_{n-1} = s') \qquad \text{(II.36)}
\end{aligned}$$

Les expressions de C_{γ_n} et $\gamma_n^{i}(\mathbf{R}_n, s', s)$ deviendraient:

$$C_{\gamma_n} = \frac{Pr(\tilde{\Lambda}_n)}{2\pi\sigma(1+e^{\tilde{\Lambda}_n})} e^{-m_a \frac{x_n^2+y_n^2+2(x_n+c_n+1)}{2\sigma^2}} \qquad \text{et}$$

$$\gamma_n^{i}(\mathbf{R}_n, s', s) = e^{m_a \frac{2(x_n i+y_n c_n)}{\sigma^2}+i\tilde{\Lambda}_n} Pr(S_n = s | u_n = i, S_{n-1} = s')$$

Annexe III

Résultats de simulation des codes turbo utilisant l'entrelaceur à double décalage cyclique

L'entrelaceur à double décalage cyclique non alterné est caractérisé par un décalage d'une part des lignes de $(i-1)D$ vers la gauche et d'autre part des colonnes de $i \times D \ mod(m)$ vers le bas seulement. Où i est le numéro de la ligne, m nombre de lignes de la matrice d'entrelacement, n nombre de colonnes de celle-ci et D est la partie entière de $\frac{n}{m}$.

Alors que l'entrelaceur à double décalage cyclique alterné est celui que nous avons déjà présenté dans le chapitre 2.

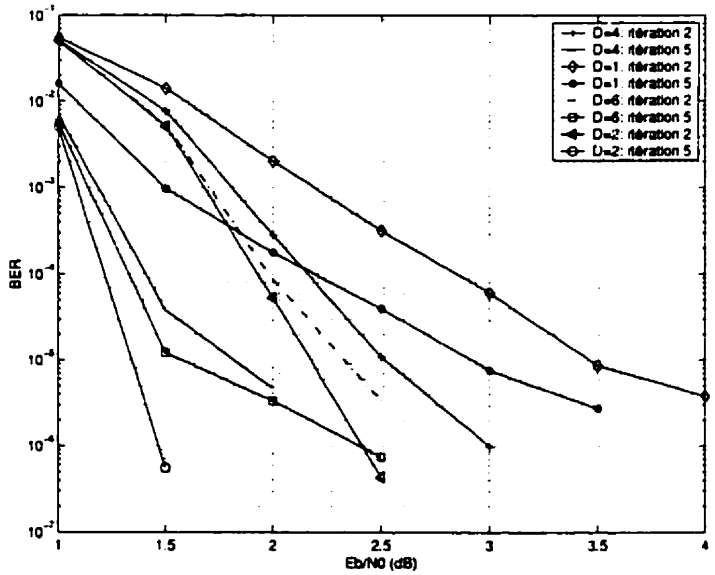


Figure III.1: Performance des codes turbo utilisant l'entrelaceur à double décalage cyclique de taille 3600 pour différentes valeurs de D dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 1/2$

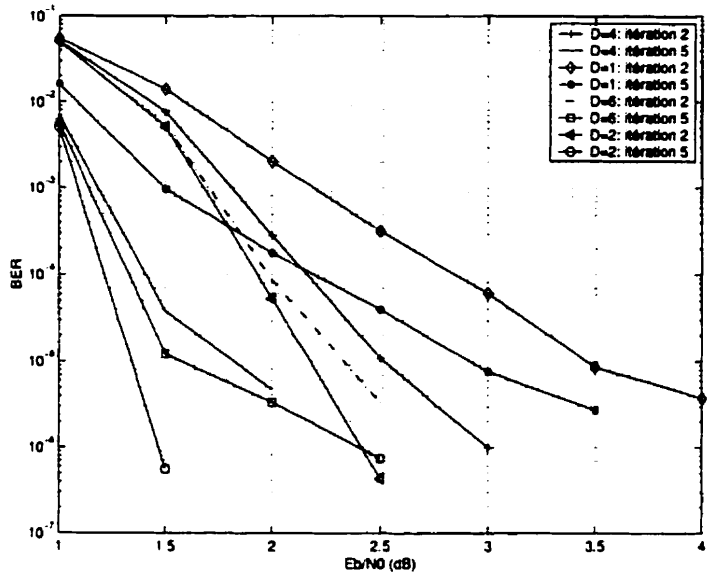


Figure III.2: Performance des codes turbo utilisant l'entrelaceur à simple décalage cyclique de taille 3600 pour différentes valeurs de D dans un canal AWGN avec $K = 5$, $G = (1, 21/37)$, $R_{gp} = 1/3$

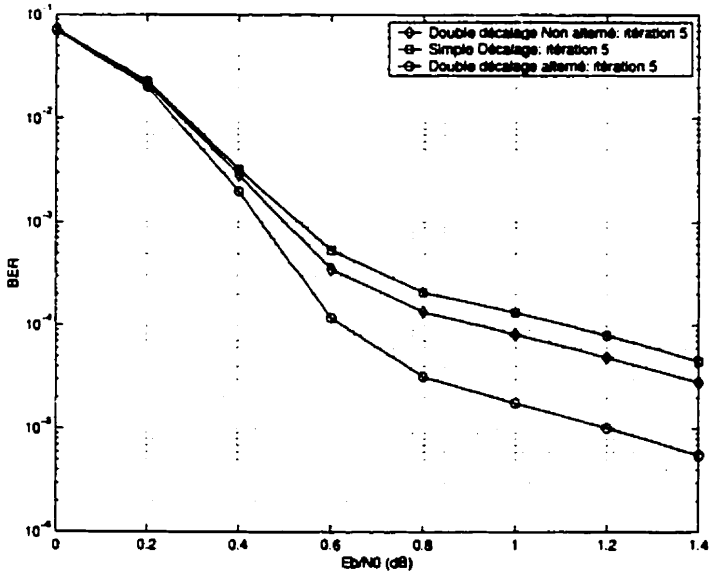


Figure III.3: Performance des codes turbo utilisant l'entrelaceur à décalage cyclique de taille 3600 dans un canal AWGN avec $K = 5$, $G = (1, 21/37)$, $R_{gp} = 1/3$

Annexe IV

Résultats de simulation des codes turbo perforés utilisant le nouveau principe du patron de perforation

IV.1 Codeurs élémentaires CRS de longueur de contrainte $K = 3$

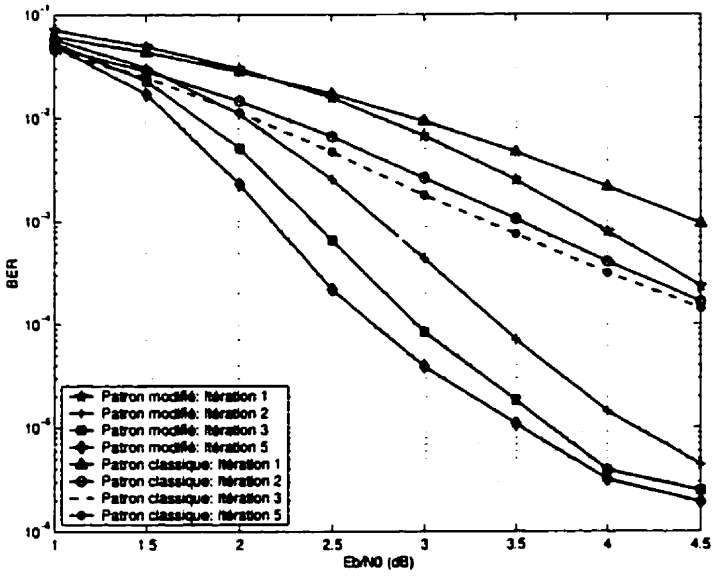


Figure IV.1: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, $R_{gp} = 3/5$ et taille d'entrelaceur aléatoire 900

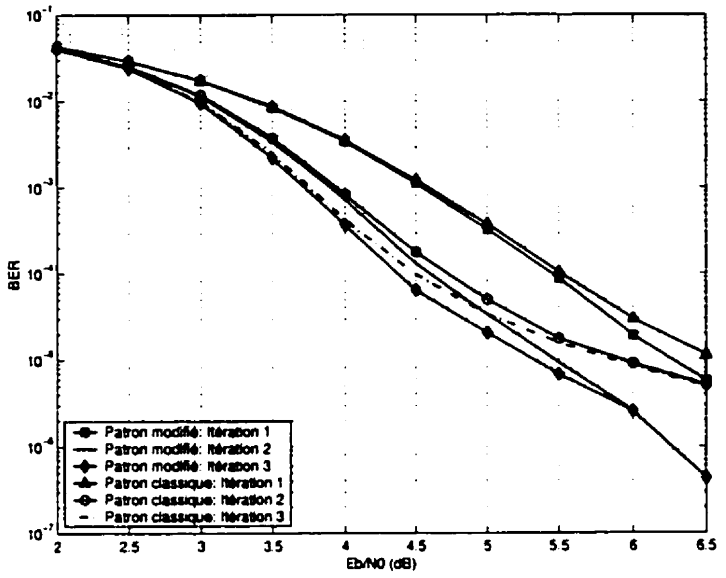


Figure IV.2: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, $R_{gp} = 10/12$ et taille d'entrelaceur aléatoire 900

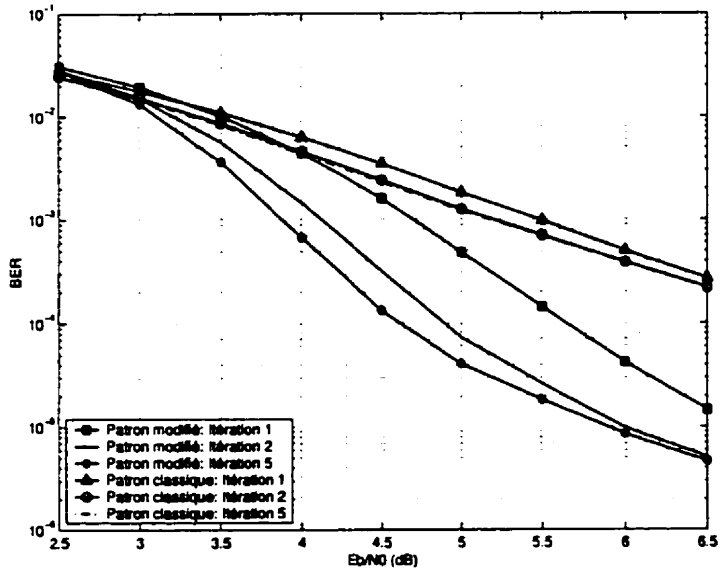


Figure IV.3: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, $R_{gp} = 12/14$ et taille d'entrelaceur aléatoire 900

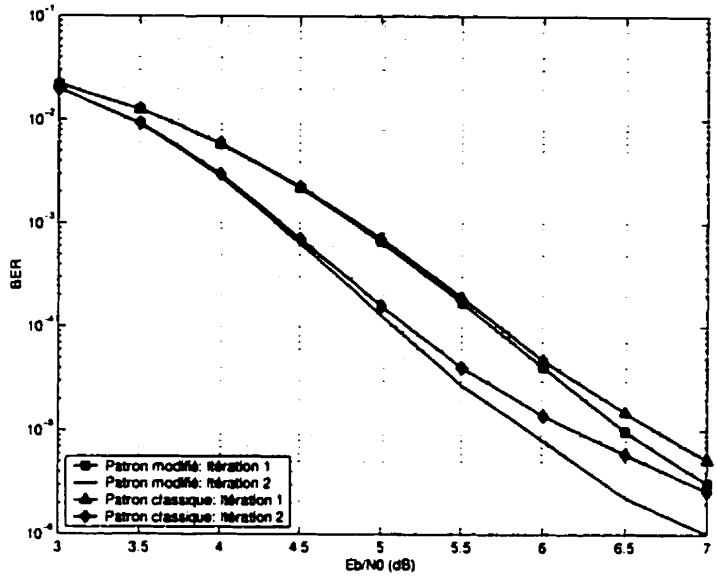


Figure IV.4: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, $R_{gp} = 16/18$ et taille d'entrelaceur aléatoire 900

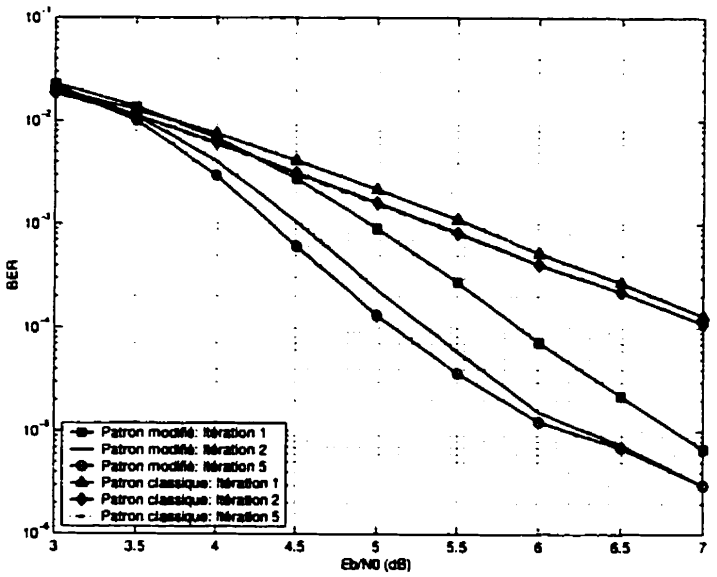


Figure IV.5: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, $R_{gp} = 18/20$ et taille d'entrelaceur aléatoire 900

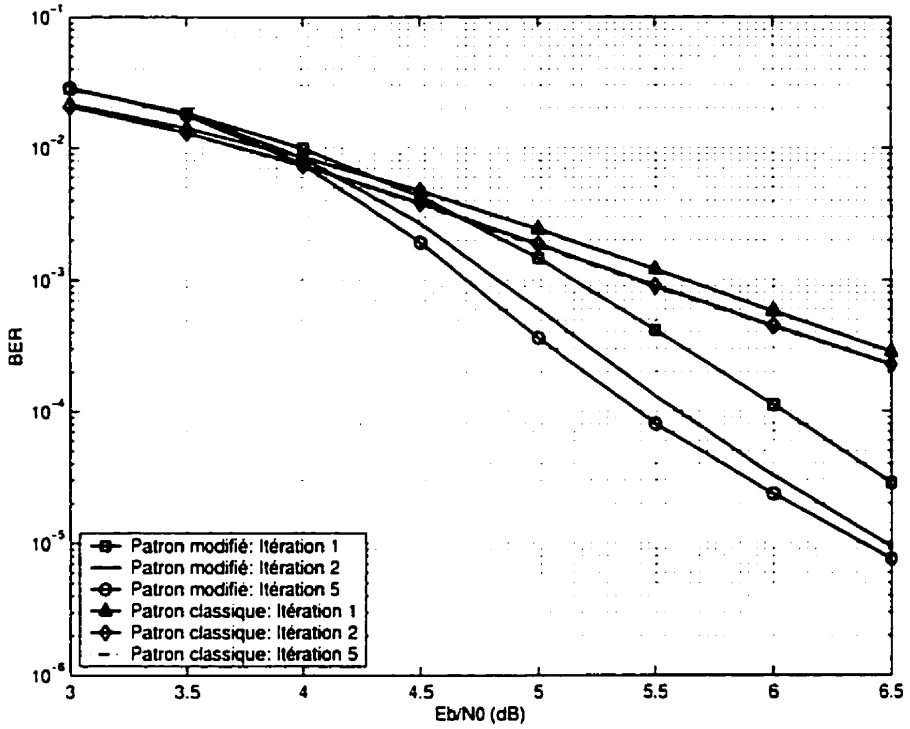


Figure IV.6: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 3$, $G = (1, 5/7)$, $R_{gp} = 24/26$ et taille d'entrelaceur aléatoire 900

IV.2 Codeurs élémentaires CRS de longueur de contrainte $K = 4$

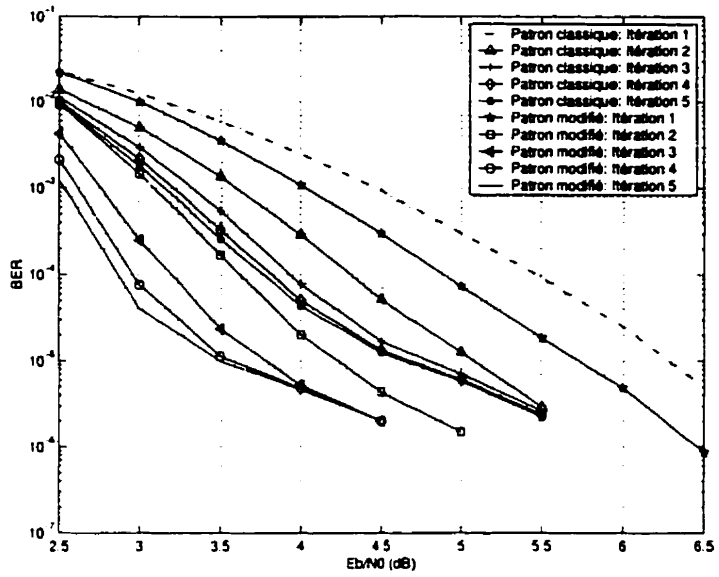


Figure IV.7: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 4$, $G = (1, 15/17)$, $R_{gp} = 6/8$ et taille d'entrelaceur aléatoire 3600

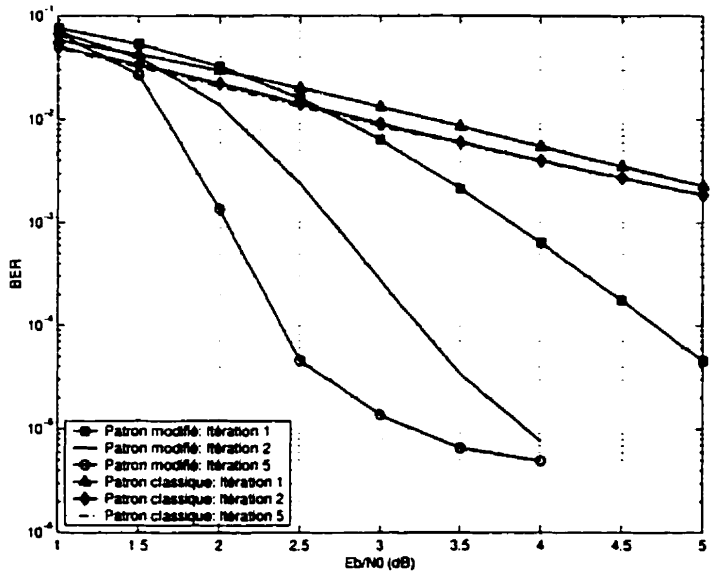


Figure IV.8: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 4$, $G = (1, 15/17)$, $R_{gp} = 4/6$ et taille d'entrelaceur aléatoire 3600

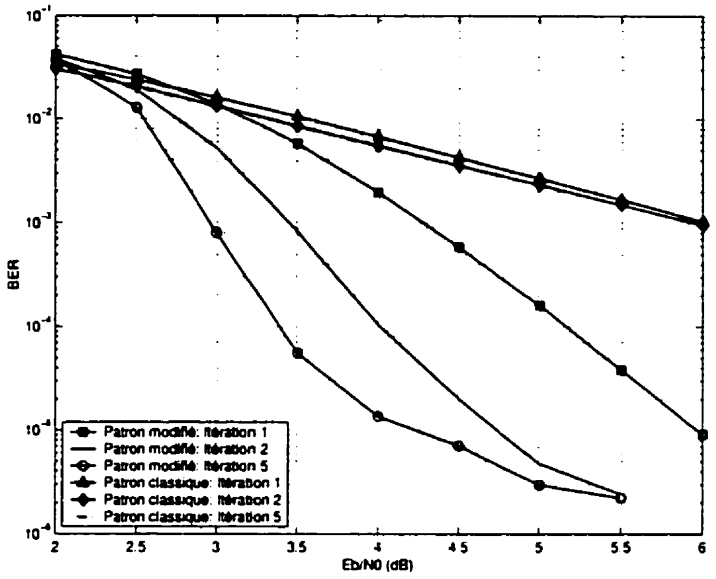


Figure IV.9: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 4$, $G = (1, 15/17)$, $R_{gp} = 8/10$ et taille d'entrelaceur aléatoire 3600

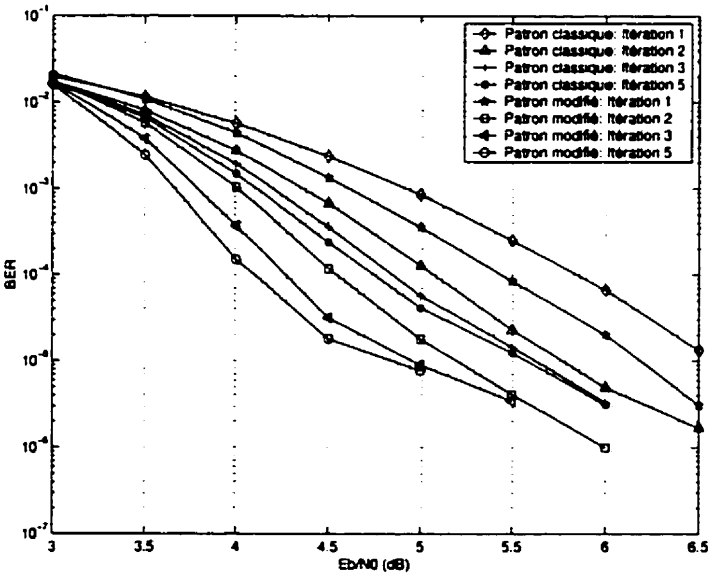


Figure IV.10: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 4$, $G = (1, 15/17)$, $R_{gp} = 14/16$ et taille d'entrelaceur aléatoire 3600

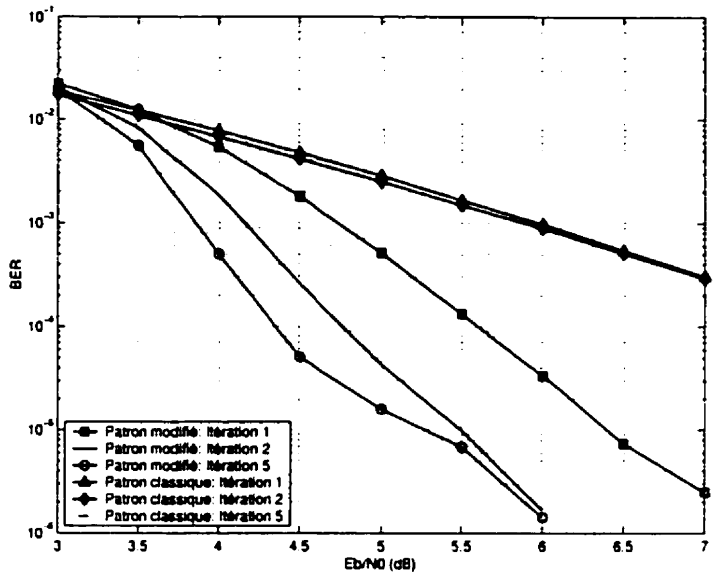


Figure IV.11: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 4$, $G = (1, 15/17)$, $R_{gp} = 16/18$ et taille d'entrelaceur aléatoire 3600

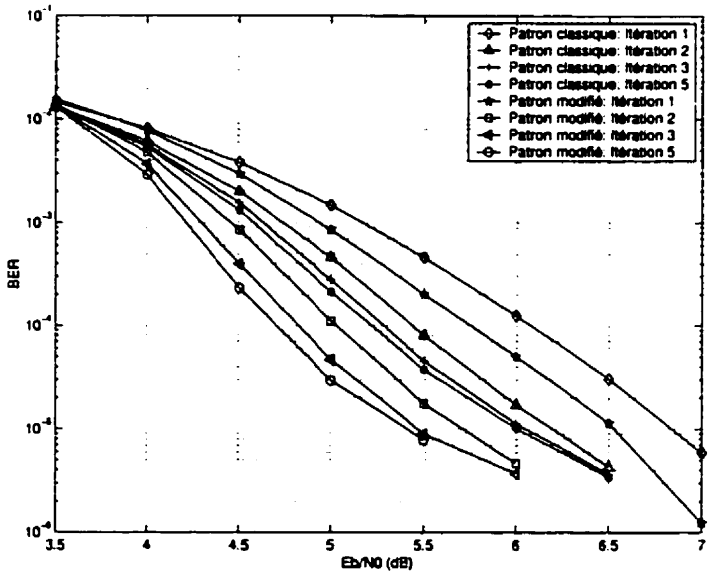


Figure IV.12: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 4$, $G = (1, 15/17)$, $R_{gp} = 22/24$ et taille d'entrelaceur aléatoire 3600

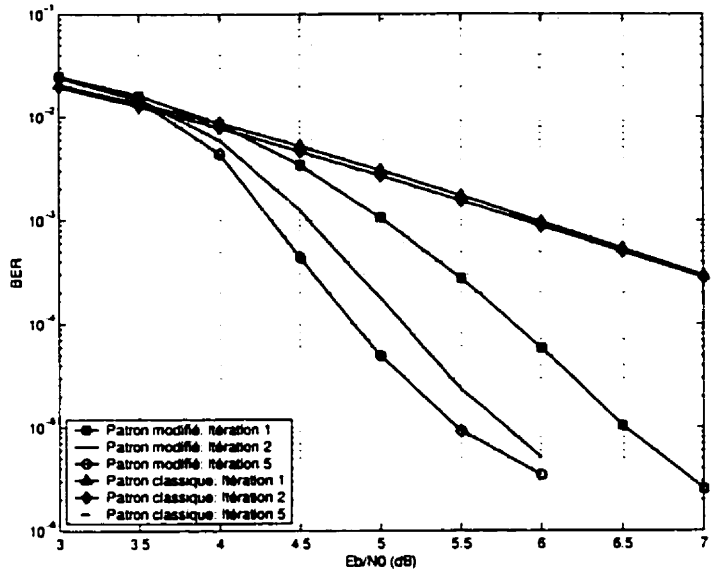


Figure IV.13: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 4$, $G = (1, 15/17)$, $R_{gp} = 24/26$ et taille d'entrelaceur aléatoire 3600

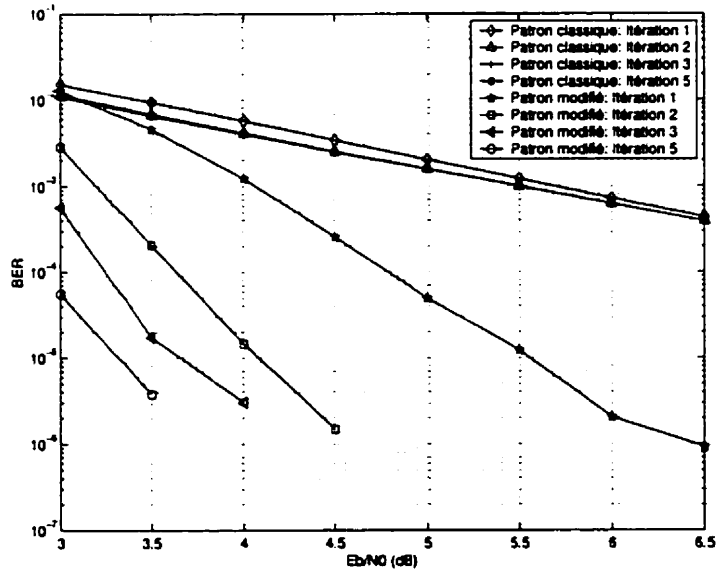


Figure IV.14: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 4$, $G = (1, 17/15)$, $R_{gp} = 7/9$ et taille d'entrelaceur aléatoire 3600

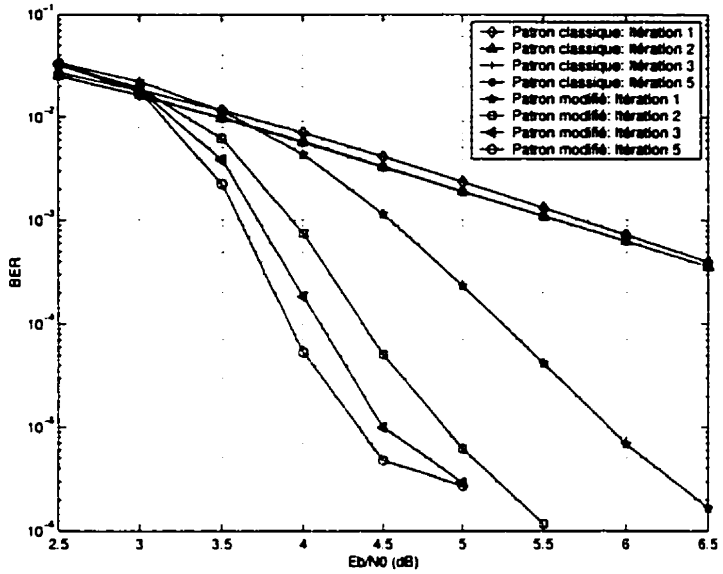


Figure IV.15: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 4$, $G = (1, 17/15)$, $R_{gp} = 14/16$ et taille d'entrelaceur aléatoire 3600

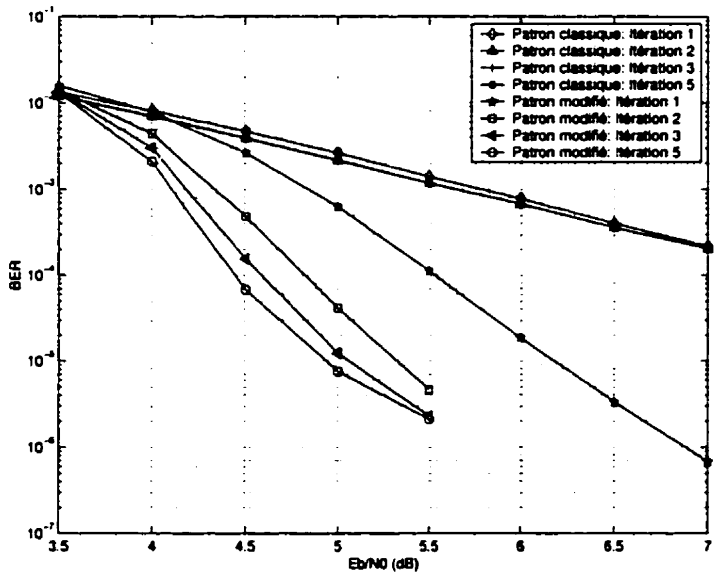


Figure IV.16: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 4$, $G = (1, 17/15)$, $R_{gp} = 21/23$ et taille d'entrelaceur aléatoire 3600

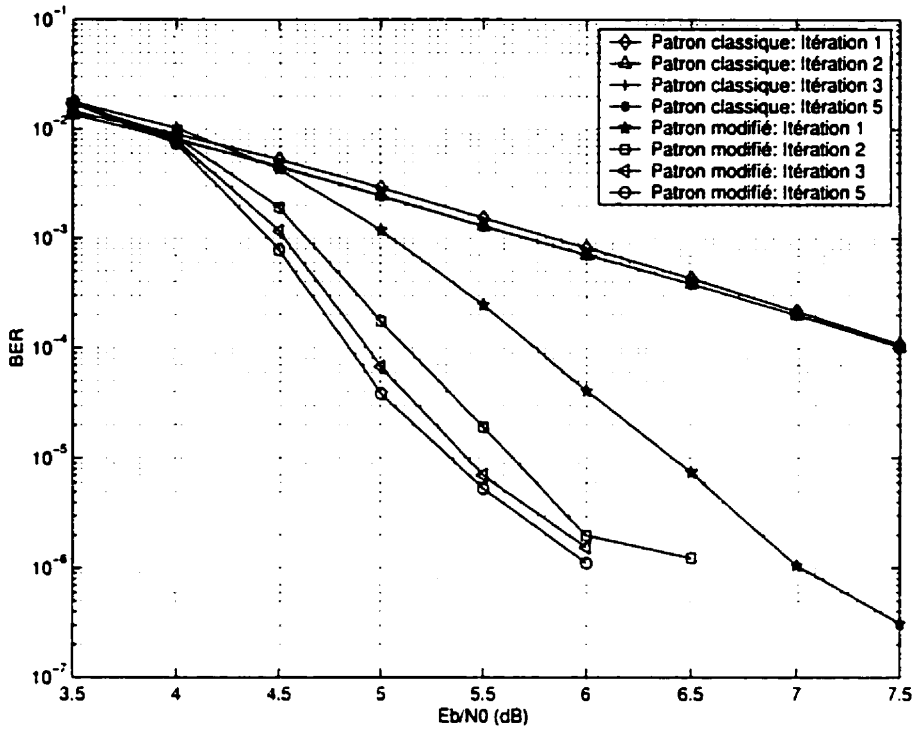


Figure IV.17: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 4$, $G = (1, 17/15)$, $R_{gp} = 28/30$ et taille d'entrelaceur aléatoire 3600

IV.3 Codeurs élémentaires CRS de longueur de contrainte $K = 5$

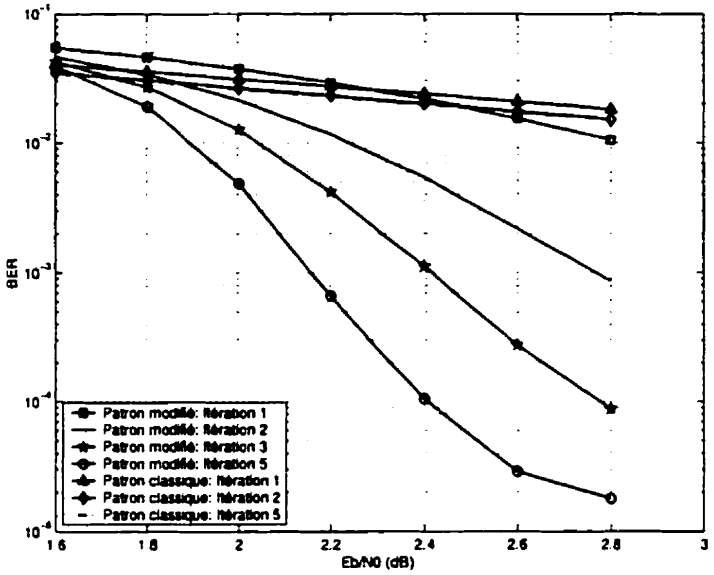


Figure IV.18: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 5$, $G = (1, 21/37)$, $R_{gp} = 5/7$ et taille d'entrelaceur aléatoire 4096

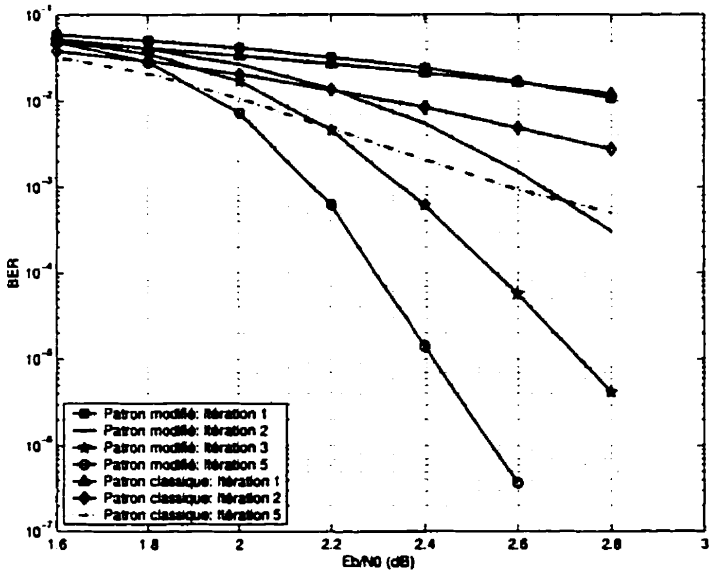


Figure IV.19: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 5/7$ et taille d'entrelaceur aléatoire 4096

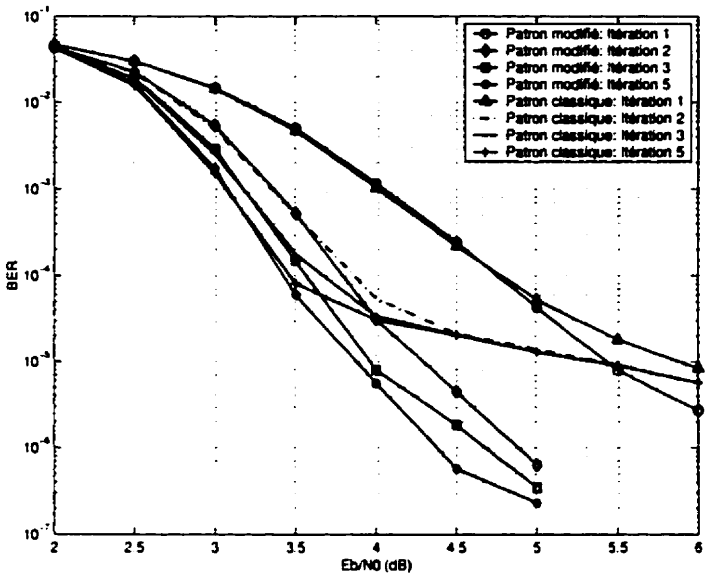


Figure IV.20: Comparaison entre l'ancienne et la nouvelle méthode de perforation pour un taux de codage $R_{gp} = 8/10$ dans un canal AWGN avec $K = 5$, $G = (1, \frac{35}{23})$, taille d'entrelaceur aléatoire = 1024

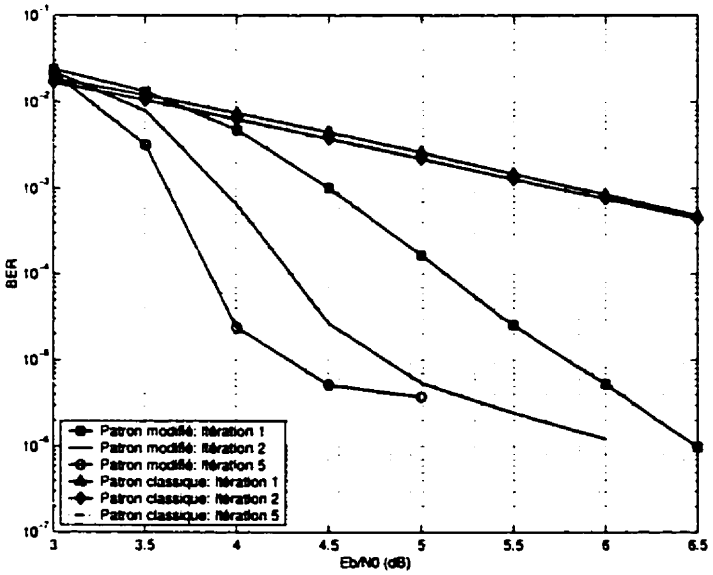


Figure IV.21: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 15/17$ et taille d'entrelaceur aléatoire 4096

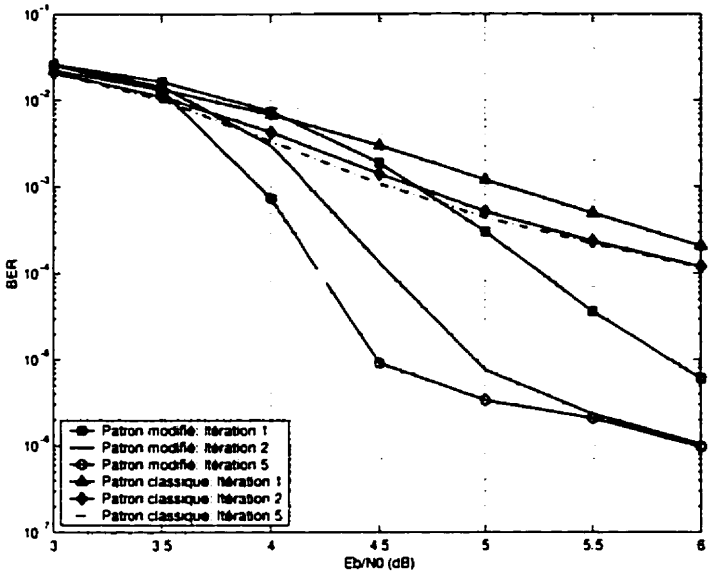


Figure IV.22: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 20/22$ et taille d'entrelaceur aléatoire 4096

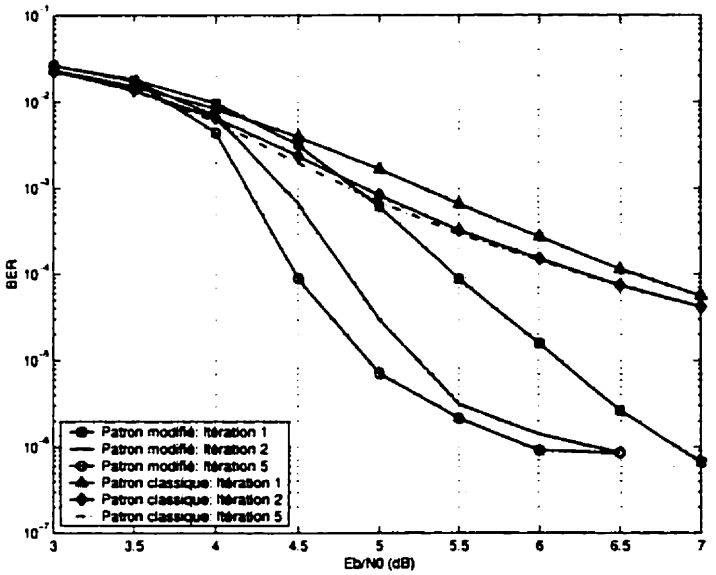


Figure IV.23: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 25/27$ et taille d'entrelaceur aléatoire 4096

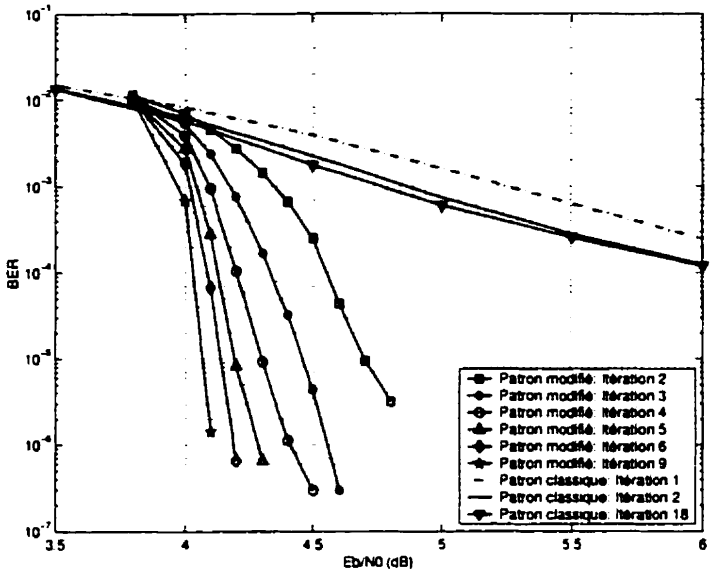


Figure IV.24: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 25/27$ et taille d'entrelaceur aléatoire 65536

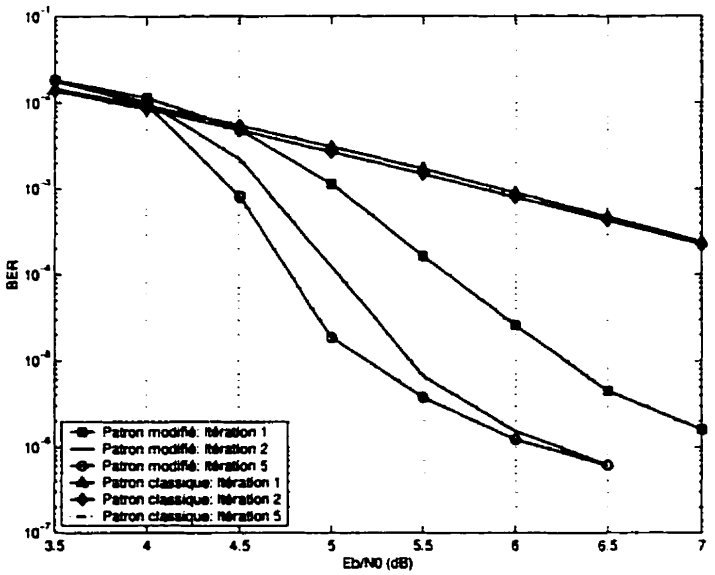


Figure IV.25: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 30/32$ et taille d'entrelaceur aléatoire 4096

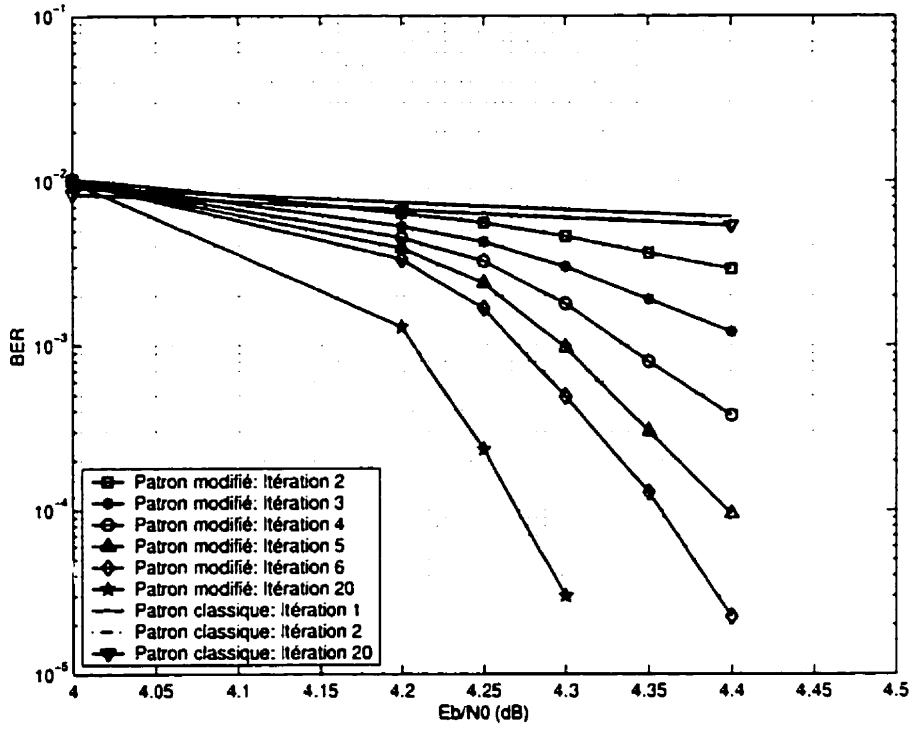


Figure IV.26: Performance des codes turbo avec les patrons de perforation classique et modifié dans un canal AWGN avec $K = 5$, $G = (1, 35/23)$, $R_{gp} = 30/32$ et taille d'entrelaceur aléatoire 65536

Annexe V

Recherche de la meilleure matrice de perforation en utilisant le principe de la nouvelle méthode proposée

L'étude pour trouver les meilleures positions des symboles retenus dans le nouveau patron de perforation est assez compliquée. Nous avons essayé de déterminer cette matrice pour $R_{gp} = 3/5$ avec des codeurs CRS caractérisés par une longueur de contrainte $K = 3$ et vecteur générateur $G = (1, 5/7)$. Les résultats de simulation illustrés aux figures V.1 et V.2 indiquent qu'il y a une légère différence de performance entre l'utilisation de la matrice **P1** ou celle de **P2**.

Toutefois, une étude dans ce sens peut être encore poursuivie et déterminer ainsi la matrice qui donne les meilleures performances pour les codes turbo.

$$\mathbf{P1} = \left(\begin{array}{ccc|ccc|ccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right) \tag{V.1}$$

$$\mathbf{P2} = \left(\begin{array}{ccc|ccc|ccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{array}\right) \tag{V.2}$$

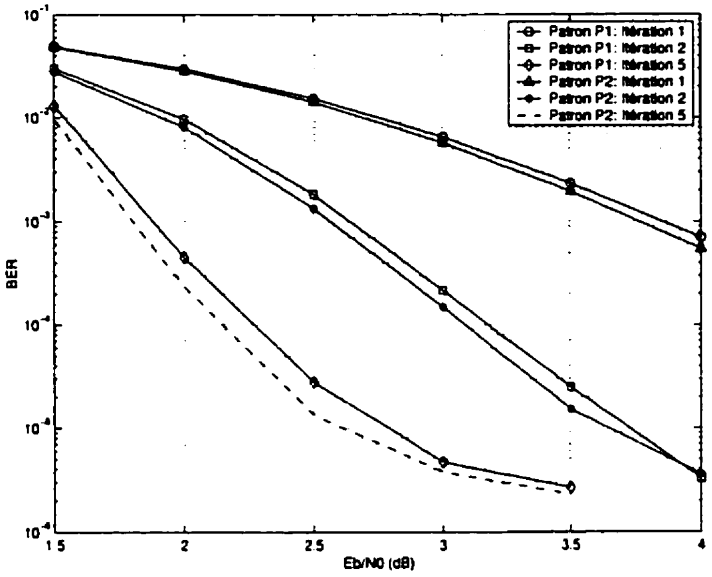


Figure V.1: Performances des codes turbo perforés suivants les patrons **P1** et **P2** avec $R_{gp} = 3/5$, $K = 3$, $G = [1, 7/5]$ et taille d'entrelaceur aléatoire 3600

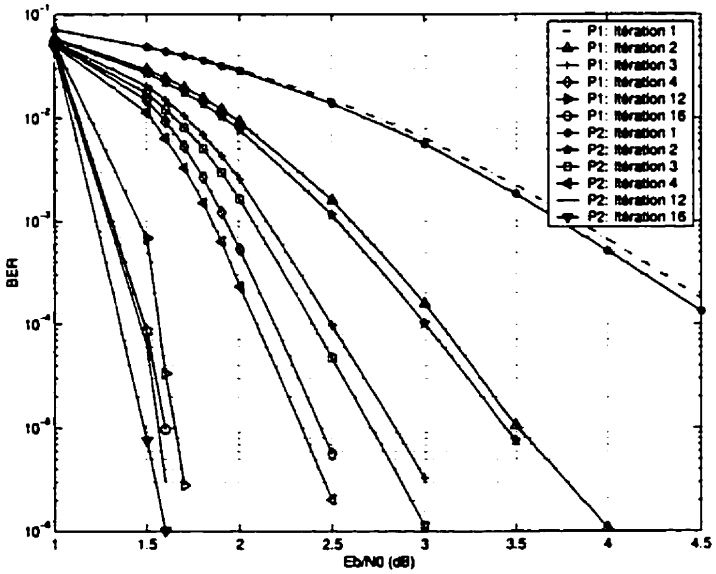


Figure V.2: Performances des codes turbo perforés suivants les patrons **P1** et **P2** avec $R_{gp} = 3/5$, $K = 3$, $G = [1, 7/5]$ et taille d'entrelaceur aléatoire 65536